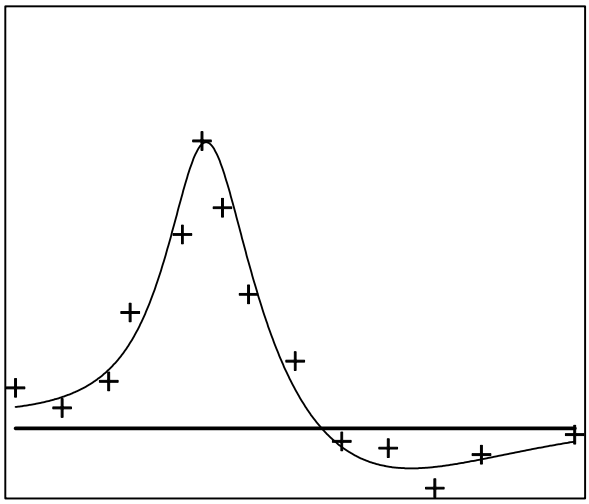
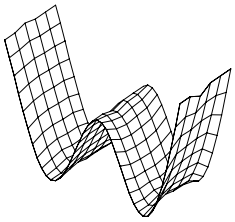
# MODFLOW-ASP

## Using MODFLOW-2000 with PEST-ASP





*Watermark Numerical Computing*

# Table of Contents

# 1. Introduction

## 1.1 General

The *observation*, *sensitivity* and *parameter estimation* processes included in MODFLOW-2000 (Harbaugh et al, 2000; Hill et al, 2000) allow MODFLOW calibration to be undertaken without the need for any external parameter estimation software. Nevertheless, there will still be many situations in which it is preferable to calibrate a MODFLOW model using PEST rather than using inbuilt MODFLOW-2000 parameter estimation functionality. Some of these situations are now discussed.

### *1.1.1 Calibration of Multiple Models*

If MODFLOW comprises part of a more complex model involving other submodels, and if it is desired that two or more of these submodels be calibrated simultaneously, then a model-independent parameter estimator such as PEST must be used if nonlinear parameter estimation techniques are to be employed in calibrating that composite model. Nowadays it is commonplace to see MODFLOW deployed in conjunction with one or more of the following types of models in the simulation of hydraulic processes which are operative throughout a study area:-

- a recharge process model (either lumped parameter or based on the Richards Equation),
- a transport model such as MT3DMS (Zheng and Wang, 1998),
- a surface water process model.

### *1.1.2 Access to Advanced Parameter Estimation and Predictive Analysis Capabilities*

PEST includes features that are not included within the *parameter estimation* process of MODFLOW-2000. These include the following.

- PEST allows upper and lower bounds to be individually placed on the values taken by parameters throughout the parameter estimation process. The methodology by which these bounds are imposed is based on a unique algorithm that lends stability to the parameter estimation process as the limits are applied. While MODFLOW-2000 allows the user to supply parameter upper and lower bounds as part of the input dataset of its *parameter estimation* process, these bounds are not enforced.
- PEST allows parameters to be linked; values of linked parameters maintain a user-specified ratio throughout the parameter estimation process.

- Observations can be collected into groups so that the contribution made by different observation types, or of observations in different areas, to the overall objective function can be monitored throughout the parameter estimation process.

- PEST includes an extremely powerful nonlinear predictive analyser. This allows the user to maximise or minimise a key model prediction while ensuring that the constraints on model parameter values imposed by the calibration process are respected.

### *1.1.3 Working with Highly Parameterised and Complex Systems*

- PEST includes a mode of operation known as "regularisation mode". This is a particularly powerful way of working with spatially complex distributed parameter systems whereby the user is able to enforce smoothness or other criteria across a model domain, while still allowing the model to be as spatially complex as it needs to be in order to simulate system behaviour.

- The user-intervention capabilities of PEST are particularly useful in situations where many parameters are being estimated simultaneously. As it advances through the parameter estimation process, PEST records information which allows rapid identification of troublesome parameters (normally insensitive parameters) which are hampering the progress of this process. These parameters can then be temporarily held at their current values while the parameter upgrade vector is calculated; alternatively, if the user suspects that the objective function has not been lowered sufficiently during any one optimisation iteration, the parameter upgrade vector can be re-calculated with one or more recalcitrant parameters held at their current values without the need to re-compute the Jacobian matrix.

- Where complex spatial parameterisation schemes are invoked to describe the distribution of some hydraulic property over the model domain, it may not be possible to define this distribution in terms of existing MODFLOW-2000 *parameter* functionality. In that case a composite model will have to be constructed, comprised of a "parameter pre-processor" (which calculates one or more MODFLOW layer property arrays on the basis of parameters which govern the distribution of that property, and writes one or more MODFLOW input files accordingly), followed by MODFLOW itself. PEST can then be used to estimate values for the parameters governing the spatial property distribution, possibly in conjunction with other, more conventional, MODFLOW parameters.

- Where key observations or predictions used in the calibration or predictive analysis process cannot be defined using MODFLOW-2000 *observation* process functionality, (for example spatially integrated or averaged water levels) a special post-processor will need to be run after MODFLOW-2000 as part of a composite model (encapsulated in a batch or script file) in order to calculate values for these key observations or predictions from traditional MODFLOW outputs. PEST can then use

the outputs of this postprocessor, together with normal MODFLOW outputs, as part of the calibration or predictive analysis process.

- PEST's parameter estimation algorithm is particularly robust. It has a proven ability to "break through" to a solution even in the face of parameter insensitivity and poor numerical behaviour of the model.

### *1.1.4 Handling of Dry Cells*

MODFLOW often encounters numerical difficulties where the water table falls or rises across the bottom of a MODFLOW layer in the course of the simulation. While these difficulties may prove troublesome in normal MODFLOW operations, they often prove disastrous when undertaking MODFLOW-based nonlinear parameter estimation. To circumvent some of the difficulties encountered in undertaking MODFLOW model calibration (or even normal MODFLOW usage), in situations where the water table crosses layer boundaries, alterations have been made to MODFLOW's normal drying/re-wetting functionality, and have been included in a special version of MODFLOW-2000 (named MF2KASP) supplied for use with PEST.

## 1.2 Calculation of Derivatives

MODFLOW-2000 has many strengths. One of its greatest strengths is its ability to calculate the Jacobian matrix internally using the sensitivity equation method, this resulting in faster, more accurate, calculation of derivatives than can be undertaken by PEST on the basis of finite parameter differences. PEST is forced to calculate derivatives using finite differences because of its model-independence. However this can be a slow process, requiring at least one model run per optimisation iteration for each adjustable parameter. Furthermore, the numerical imprecision caused by this process can have a deleterious effect on the overall parameter estimation process in difficult situations.

Versions of PEST from 5.0 onwards can use derivatives which are actually calculated by the model if these are available. When calibrating MODFLOW-2000 using PEST, a special version of MODFLOW-2000 (called MF2KASP) must be used. This version of MODFLOW-2000 has been slightly modified to write the MODFLOW-calculated sensitivity matrix to a file. PEST reads this file once during every optimisation iteration, incorporating the information contained in this file into its Jacobian matrix. When PEST is calibrating a complex model, some derivatives can be supplied by MODFLOW itself in this manner, while others can be calculated by PEST using finite differences. This results in a system of unprecedented power and flexibility that allows the calibration of complex models for which nonlinear parameter estimation techniques would not have previously been considered.

In many cases, calibration of a MODFLOW-2000 model using PEST will not be as efficient as calibrating a MODFLOW-2000 model using MODFLOW-2000's own inbuilt *parameter estimation* functionality. This will apply particularly to smaller models, or to models where only a few parameters require estimation. This is because there are

some overheads involved in using PEST with MODFLOW-2000. These include the fact that MODFLOW-2000 must be re-started each time it is run by PEST, which is at least twice per optimisation iteration. In contrast, when MODFLOW-2000 is calibrated by MODFLOW-2000, the MODFLOW-2000 executable program, and the data which it uses, is loaded into memory only once. This, and other, overheads will be more noticeable when calibrating a small model than when calibrating a large model due to the smaller relative time that is devoted to actual mathematical computation in comparison to data input/output when the model is small.

## 1.3 The MODFLOW-2000 to PEST Translator

To simplify use of PEST with MODFLOW-2000, a special utility program named MF2PEST has been developed. MF2PEST generates a set of PEST input files on the basis of a set of MODFLOW-2000 input files. Thus once a set of MODFLOW-2000 input files have been prepared by any MODFLOW graphical user interface, MF2PEST can be used to create a set of PEST input files in which the same parameters are estimated as those identified for estimation in the MODFLOW-2000 *parameter estimation* process input file. Once this has been done, PEST can then be used to calibrate the MODFLOW model. When the parameter estimation process is complete, a small utility program named PAR2SEN can be used to transfer optimised parameter values to an existing MODFLOW-2000 *sensitivity* process input file.

In its simplest mode of operation MF2PEST supplies default values for all PEST control variables as it writes the PEST input dataset. However, if you wish, you can provide MF2PEST with your own choice of value for any PEST control variable (and for a number of special MF2KASP variables). Furthermore, you can ask PEST to run in predictive analysis or regularisation modes rather than in its traditional parameter estimation mode. You can also collect parameters and/or observations into groups on the basis of their MODFLOW-2000 names. Many other options for constructing the PEST input dataset are also available through MF2PEST.

## 1.4 Modified Version of MODFLOW-2000

As mentioned above, a special version of MODFLOW-2000 named MF2KASP is provided with MF2PEST. MF2KASP has been slightly modified from the MODFLOW-2000 program provided by the United States Geological Survey. These alterations are as follows:-

1. MF2KASP writes a “derivatives file” from which PEST can read the sensitivity of every model output for which there is a complimentary field measurement to every adjustable parameter. Versions of PEST from 5.0 onwards can partly or wholly fill their Jacobian matrix on the basis of the contents of a file such as this.

2. Cells in the bottom layer of a single or multi-layered model can be prevented from drying out.

3. The head in a cell that is declared as "dry" can be assigned a value equal to the elevation of the bottom that cell.

4. If an observation bore is located in a cell that is declared as "dry", sensitivities of observations pertaining to that bore with respect to all adjustable parameters are assigned a value of 0.0

5. MODFLOW can be prevented from terminating execution upon solution convergence failure.

6. MODFLOW file output can be dramatically reduced in order to lessen the computational burden of running MODFLOW many times under the control of PEST during the parameter estimation process.

7. The mechanism by which MODFLOW hydraulic property arrays are calculated from pertinent MODFLOW-2000 parameters can be altered to better accommodate the use of pilot points as a spatial parameterisation device.

These alterations are discussed in detail in later sections of this document.

It should be noted that in spite of the fact that MODFLOW's "drying/re-wetting" functionality has been altered to mitigate its deleterious effect on the parameter estimation process, the occurrence of dry cells should be avoided at all costs. Even if an observation bore does not lie within a dry cell, MODFLOW's handling of drying and re-wetting conditions introduces small discontinuities into the relationships between parameters and model outputs. This causes errors in the calculation of derivatives; these errors will almost certainly have a negative effect on the parameter estimation process.

## 1.5 Alternative Calibration Strategy

Considerable software support for the construction of PEST input files for the calibration of MODFLOW and MT3DMS (either together or separately) is available through the "Groundwater Data Utilities" produced by Watermark Numerical Computing (2001).Use of some of these utilities is now redundant, thanks to the software described in this document. However in other instances these utilities will be as useful as ever, particularly when calibrating MODFLOW in conjunction with another model such as MT3DMS. Functionality available through these utilities includes the following:-

- spatial and temporal interpolation of MODFLOW head and drawdown output files to the times and sites of borehole head and drawdown measurements;

- spatial and temporal interpolation of MODFLOW budget output files to the times and sites of head-dependent flow measurements (for use with packages such as *drain*, *river*, *general head boundary* etc,),

- spatial and temporal interpolation of MT3DMS concentration files to the sites and times of borehole concentration measurements,

- manipulation and pre-processing of MODFLOW and MT3DMS two-dimensional input arrays,

- automatic generation of PEST control and instruction files prior to a MODFLOW or MT3DMS calibration run,

- use of pilot points as a spatial parameterisation device complemented by spatial interpolation based on simple or ordinary kriging,

- modification of a pilot-points-based PEST input file in order to introduce geostatistically-based regularisation constraints into the inversion process,

- modification of an existing MODFLOW-2000 input dataset in which zone-based parameters are replaced by pilot-point-based parameters.

All of the functionality available through these utilities for the calibration of MODFLOW 88 and MODFLOW 96 models is in no way diminished when applied to model calibration using MODFLOW-2000, either in conjunction with, or independently of, the *observation, sensitivity* and *parameter estimation* processes encapsulated in the latter package.

# 2. Basic MF2PEST Usage

## 2.1 General

Because MF2PEST (the MODFLOW-2000 to PEST dataset translator) supplies sensible default values for just about all of PEST's input variables, it is a particularly easy program to use if these default values are suitable for your particular problem (which is true in the vast majority of cases). This section discusses how to use PEST with MODFLOW-2000 under these simple circumstances. The next section discusses more advanced MF2PEST and PEST usage.

## 2.2 MF2PEST Input Control File

Like PEST, MF2PEST requires an input control file. And, like PEST, the name of the input control file is supplied on the command line. Like the PEST control file, the MF2PEST input control file is subdivided into sections. Each of these sections begins with a header, this header being comprised of the "*" character followed by the name of the section that follows. In MF2PEST's basic mode of operation only one section is required in its input control file, viz. the *modflow control* section. Other sections (which can be omitted if they are not needed) are named *pest control*, *predictive analysis*, *regularisation*, *modflow observations*, *modflow parameters*, *parameter groups* and *general*. These sections can be supplied in any order in the MF2PEST input control file.

Formatting of an MF2PEST input control file is much more flexible than that of a PEST control file. Most lines within an MF2PEST input control file should contain two entries. The first is the name of an MF2PEST input variable; the second is the value of that variable. Within each section of the MF2PEST input control file, MF2PEST variables can be listed in any order. Furthermore, any variable can be omitted if you are happy with the default value supplied by MF2PEST for that variable.

Data supplied in an MF2PEST input control file is case-insensitive. Thus section headers and MF2PEST variable names can be supplied in upper or lower case (or a mixture of both).

An MF2PEST input control file, in its most basic form, is illustrated in Figure 1. In many cases of MF2PEST usage there is no need for an MF2PEST input control file to be any more complex than this.

```
* modflow control
 MODNAMFILE d:\asps\test\tc1.nam
 MODFLOWCOM mf2kasp
```

**Figure 1. The most basic form of an MF2PEST input control file.**

Values are supplied for only two MF2PEST variables in the MF2PEST input control file illustrated in Figure 1, viz. for the variables MODNAMFILE and MODFLOWCOM. The first of these is the name of the MODFLOW name file pertaining to the current MODFLOW-2000 dataset, while the second is the command that PEST should use to run MODFLOW-2000. Note the following:-

- The MODFLOW name file can be prefixed by a full directory name if desired. If there is no directory prefix, it is assumed that the file resides in the current working directory (ie. the directory from which MF2PEST is run).

- The name of the MODFLOW-2000 executable program can be prefixed by a full directory name if desired. If there is no directory prefix, it is assumed that the pertinent executable program resides either in the current working directory, or in a directory cited in the PATH environment variable.

- The MODFLOW-2000 executable program must be a special version of MODFLOW-2000 (named MF2KASP) supplied with MF2PEST. As was discussed above, this version of MODFLOW-2000 is slightly altered from the USGS version of MODFLOW-2000 such that (a) it is capable of recording parameter sensitivities to a "derivatives file" in the format expected by PEST, and (b) it handles drying/re-wetting conditions in a slightly different way to the normal MODFLOW-2000.

## 2.3 Running MF2PEST

MF2PEST is run using the command:-

```
mf2pest infile
```

where *infile* is the name of an MF2PEST input control file (use quotes if this filename contains spaces). Make sure that MF2PEST resides in a directory cited in the PATH environment variable. If it does not, prefix the above command with the name of the directory in which the MF2PEST executable program *mf2pest.exe* resides.

## 2.4 What MF2PEST Does

If MF2PEST is run using the input control file depicted in Figure 1, it carries out the activities set out below. Operation of MF2PEST if supplied with a more complex input control file is very similar, the main difference being that user-supplied values, rather than MF2PEST default values, are given to variables cited in the PEST control file built by MF2PEST; see Section 3 of this document for further details.

### *2.4.1 MODFLOW Input Files*

After it has read its input control file, MF2PEST turns its attention to the MODFLOW-2000 input dataset (which must have been prepared prior to running MF2PEST). First MF2PEST reads the MODFLOW name file provided as the value of the MODNAMFILE variable cited in the MF2PEST input control file. If, on reading this file, MF2PEST discovers that the MODFLOW-2000 *parameter estimation* process is not active, it stops execution with an appropriate error message. Otherwise it next reads the MODFLOW-2000 *sensitivity* process input file, from which it ascertains the names of all adjustable parameters, their initial values, their suggested upper and lower bounds, and whether they are log-transformed or not in the inversion process.

### *2.4.2 Template Files*

Next MF2PEST writes two PEST template files. Each of these template files pertains to a MODFLOW-2000 *sensitivity* process input file for the current case. Whenever PEST runs MODFLOW-2000 during the parameter estimation process it first supplies MODFLOW-2000 with a set of parameter values to use on its current run; these are supplied through its *sensitivity* process input file. Two template files rather than one are required because PEST directs MODFLOW to use a different *sensitivity* process input file depending on whether it is being run in order to calculate derivatives required to fill the Jacobian matrix, or whether it is being run in order to test the efficacy of a new parameter set in lowering the objective function. PEST's capacity to run a different form of the model depending on its current task was introduced with version 5.0 of PEST (ie. PEST-ASP), and is discussed in detail in the documentation for that version.

The default names for the two PEST template files written by MF2PEST are *modsen1.tpl* and *modsen2.tpl*. The default names for the MODFLOW-2000 *sensitivity* process input files to which they correspond are *modsen1.sen* and *modsen2.sen*. MODFLOW-2000 is directed to these files through a new MODFLOW name file which MF2PEST will write shortly.

Note that parameter names cited in the template files written by MF2PEST are the same parameter names as those supplied in the MODFLOW-2000 *sensitivity* process input file. Thus parameter names used by PEST and MODFLOW-2000 are identical.

### *2.4.3 MODFLOW Observation Files*

After it has written the two template files, MF2PEST reads the MODFLOW-2000 global *observation* process input file, as well as the *observation* process input files for all MODFLOW packages for which there are measurements to be used in the calibration process. In generating the PEST input dataset, MF2PEST uses the same observation names as those used by MODFLOW-2000 if each of these names is unique. If the names are not unique, MF2PEST modifies them; see Section 2.12 for further details. MF2PEST calculates observation weights for the use of PEST on the basis of pertinent information supplied in these *observation* process input files. You should note carefully that *weights*

*used by PEST are the square roots of those used by MODFLOW-2000 (PEST squares them internally)*. Alternatively, if an observation covariance matrix is supplied in a MODFLOW *observation* process input file, PEST transfers that matrix to an appropriate PEST-compatible observation covariance file, and cites the name of that file in the PEST control file which it generates.

When PEST runs MODFLOW-2000 to test the efficacy of a new parameter set in lowering the objective function (as distinct from running MODFLOW-2000 to calculate derivatives to fill the Jacobian matrix), it reads the model-generated equivalents to observations from the MODFLOW-2000 "unweighted simulated equivalents to observations" file (ie. the "*_os*" file); this file is written by the MODFLOW-2000 *observation* process. The filename base of this file is supplied as the MODFLOW-2000 OUTNAM variable, found on the global *observation* process input file. If this variable is set to "none" in the MODFLOW-2000 input dataset for the current case, indicating that the "*_os*" file will not be written, MF2PEST ceases execution with an appropriate error message.

### *2.4.4 Instruction File*

MF2PEST next writes an instruction file by which MODFLOW-calculated observation equivalents can be read from the "unweighted simulated equivalents to observations" (ie. "*_os*") file written by MODFLOW-2000. The MF2PEST default name for this instruction file is *modobs.ins*.

### *2.4.5 Model Batch Files*

Two model batch files are required; PEST runs one of these batch files as the "model" when testing a new parameter set, and the other when running MODFLOW-2000 to obtain derivatives. MF2PEST provides a default name of *modrun1.bat* for the first of these files and *modrun2.bat* for the second. Each of these files contains a single line. For the first it is:-

```
MODFLOWCOM < pmodrun1.in
```

while for the second it is:-

```
MODFLOWCOM < pmodrun2.in
```

In the above command *MODFLOWCOM* is replaced by the actual value of this MF2PEST input variable; recall that this is the name of the MODFLOW-2000 executable program to be run by PEST (ie. MF2KASP). Files *pmodrun1.in* and *pmodrun2.in* cited in the above commands each contain information that would normally be supplied to MODFLOW-2000 through the keyboard, viz. the name of a MODFLOW-2000 name file. File *pmodrun1.in* contains the name of the name file that PEST wishes MODFLOW-2000 to use when it is run purely for the purpose of calculating observation equivalents on the basis of an upgraded parameter set. File *pmodrun2.in* contains the name of the name file that PEST wishes MODFLOW-2000 to use when calculating

parameter derivatives. Each of these name files contains an entry which activates the MF2KASP "ASP process", a special process added to the normal MODFLOW-2000 processes in order to enhance its interaction with PEST. However the two name files differ in the following ways:

1. The "observations only" name file informs MODFLOW-2000 that the name of its *sensitivity* process input file is *modsen1.sen* (ie. if the MF2PEST default name for this file is used, as will happen for the MF2PEST input control file depicted in Figure 1). However the "derivatives only" name file informs MODFLOW-2000 that the name of its *sensitivity* process input file is *modsen2.sen* (if the default name is used).

2. The "observations only" name file directs MODFLOW to read an "ASP input file" (see Section 5 of this document) named *setting1.asp* (ie. if the MF2PEST default name is used for this file). The "derivatives only" name file, however, directs MODFLOW to read an ASP input file named *setting2.asp* (if the default name is used). The latter ASP input file activates the MODFLOW-2000 PEST interface whereby the modified version of MODFLOW (ie. MF2KASP) records sensitivities to a special file for the use of PEST.

3. In the "observations only" name file, the MODFLOW-2000 *parameter estimation* process is disabled by "commenting out" the line of the original MODFLOW-2000 name file which contains the name of the *parameter estimation* process input file. In the *sensitivity* process input file cited in this name file (ie. *modsen1.sen*), the MODFLOW-2000 variable ISENALL is set to -1 to prevent calculation of parameter sensitivities. The parameter estimation process is also disabled in the "derivatives only" name file. However in the *sensitivity* process input file cited in this name file (ie. *modsen2.sen*) ISENALL is set to 0 so that sensitivities are calculated for all adjustable parameters.

### *2.4.6 Prior Information*

After writing the model batch files, as well as files *pmodrun1.in* and *pmodrun2.in*, MF2PEST opens the MODFLOW-2000 *parameter estimation* process input file in order to read any prior information that may reside in this file. If any prior information is present, MF2PEST calculates weights to be used by PEST for each item of prior information on the basis of pertinent variables supplied in this file. Also, if there are any prior information items present for which a covariance matrix is supplied, MF2PEST transfers the covariance matrix to an appropriate PEST-compatible observation covariance file. In the latter case PEST also assigns names to the pertinent prior information items; PEST needs to assign these names itself because prior information items with which a covariance matrix is associated are not provided with names in a MODFLOW-2000 *parameter estimation* process input file.

When comparing observations and prior information written to the PEST control file by MF2PEST with those recorded in the original MODFLOW-2000 dataset, bear in mind that weights used by PEST are the square roots of those used by MODFLOW-2000

(PEST squares them internally). Also keep in mind that PEST log-transforms parameters internally to base 10 whereas MODFLOW-2000 log-transforms them to base *e*. Weights calculated by MF2PEST for the use of PEST from the information supplied with MODFLOW-2000 prior information are adjusted on this basis. Nevertheless, it is a good idea to check the weights assigned to all prior information items on the PEST control file written by MF2PEST, just to make sure that you are happy with them.

*Note that, at the time of writing this document, there seems to be some discrepancy between the way that prior information weights are calculated by MODFLOW-2000 and the description of the calculation methodology provided in the MODFLOW-2000 manual where parameters are logarithmically transformed. Where this discrepancy exists, MODFLOW and PEST will not give the same parameter estimates in cases where prior information exists for log-transformed parameters. Once again, the user is advised to check prior information weights on the PEST control file written by MF2PEST to ensure that these are what he/she intends them to be. If not, edit the PEST control file directly, adjusting weights to the desired values, before running PEST.*

### *2.4.7 The PEST Control File*

Finally, having now read all of the information which it requires, MF2PEST writes the PEST control file. The default name for this file is *pestrun.pst*. Note the following:-

- MF2PEST's default condition is to write a PEST control file for which PEST is run in parameter estimation mode. If you wish PEST to run in predictive analysis or regularisation modes, you must supply MF2PEST with appropriate information on its input control file, or you must alter the PEST control file written by MF2PEST accordingly.

- Default values are provided by MF2PEST for all of the variables listed in the *control data* section of the PEST control file. If these are not suitable for the current case, alter them before running PEST, or provide alternative values in the MF2PEST input control file (see the next section).

- Settings governing the calculation of derivatives are such that PEST expects these to be calculated by the model itself (in this case MODFLOW-2000) rather than by PEST using finite parameter differences. When the model is run for the purpose of derivatives calculation, PEST expects to find these derivatives in a file named *modflow.der*.

- All parameters are assigned to a parameter group named *general*. Default settings for the finite-difference calculation of derivatives are provided for this group. However PEST will ignore these settings as derivatives will be calculated by the model itself rather than by PEST.

- All parameters that are log-transformed in the MODFLOW-2000 input dataset are designated as log-transformed in the PEST control file. Similarly, untransformed and

fixed parameters in the MODFLOW-2000 input dataset are designated as untransformed and fixed respectively in the PEST control file which is written by MF2PEST.

- Initial parameter values supplied in the MODFLOW-2000 *sensitivity* process input file are transferred directly to the PEST control file.

- Upper and lower bounds recorded on the MODFLOW-2000 *sensitivity* process input file are transferred directly to the PEST control file. **Note, however, that MODFLOW-2000 has no capacity to enforce these bounds, whereas PEST does. If you do not wish PEST to enforce these bounds, set them much wider.**

- If a parameter is log-transformed, or if its upper and lower bounds are of the same sign in the MODFLOW-2000 *sensitivity* process input file, the parameter is designated as factor-limited (PEST input variable PARCHGLIM). However if a parameter is not log-transformed and if its bounds are of opposite sign, the parameter is designated as relative-limited in the PEST control file written by MF2PEST.

- Observations are assigned to observation groups whose names reflect the MODFLOW-2000 *observation* package to which they pertain. Default observation group names are *head*, *ghb*, *drain*, *drt*, *river*, *stream*, *adv* and *const_head*. Prior information for which a covariance matrix is not supplied is assigned to an observation group named *prior_info*; prior information for which a covariance matrix is provided is assigned to the observation group *prior_cov*.

## 2.5 Checking the PEST Input Dataset

After MF2PEST has written its suite of PEST and MODFLOW-2000 input files, you are nearly ready to run PEST (version 5.0 or later). But first the MF2PEST-generated PEST input dataset must be checked for correctness and consistency using PESTCHEK (version 5.0 or later). Even though PEST's input dataset was prepared by MF2PEST, it is still possible for it to contain inconsistencies. For example, the upper bound for a parameter supplied in the MODFLOW-2000 *sensitivity* process input file may be lower than the lower bound supplied for the same parameter; the initial parameter value may exceed its upper bound or may be smaller than its lower bound; etc. While MF2PEST checks for obvious problems in the MODFLOW-2000 input dataset as it reads it, it does not check for every possible error or inconsistency because an "everything checker" in the form of PESTCHEK already exists. Hence, as is normal practice before running PEST, PESTCHEK should be run in order to carry out all of PEST's "pre-flight checks".

## 2.6 MF2K Settings

As is explained in Section 5 of this document, modifications to the version of MODFLOW-2000 supplied with MF2PEST (ie. MF2KASP) extend beyond those required to provide PEST with MODFLOW-calculated derivatives with respect to adjustable parameters. A number of other alterations pertain to the manner in which

MODFLOW-2000 handles the situation where the water table falls below the bottom of a layer. See Section 5 for full details.

Settings which govern the operation of its enhanced functionality must be supplied to MF2KASP in an "ASP input file". MF2PEST writes this file itself. If supplied with the MF2PEST control file shown in Figure 1, MF2PEST provides default values for all MF2KASP settings when it writes the ASP input file. However, as is described in Section 3 of this document, MF2PEST can be directed to supply different settings through the inclusion of suitable entries in its input control file.

## 2.7 Running PEST

As is recorded in the documentation to version 5.0 or later of PEST, PEST's ability to use model-calculated derivatives is only available through the "single window" version of PEST; it is not available through Parallel PEST. (However it is available through WinPEST.) To run PEST on the basis of the default PEST control filename provided by MF2PEST, use the command:-

```
pest pestrun
```

You may wish to compare the results of a PEST run with those of a MODFLOW-2000 *parameter estimation* run based on the same input dataset. When doing this, remember that it is not a foregone conclusion that PEST will calculate the same parameter values as MODFLOW-2000 does because of the nonuniqueness of the parameter estimation problem in many situations. However PEST should lower the objective function to the same extent. The objective functions calculated by MODFLOW-2000 and PEST are, in general, directly comparable. However, in many situations objective functions calculated by the packages will differ for the reasons described below. (They will also differ where prior information is supplied for log-transformed parameters as discussed above.)

For small models MODFLOW-2000 will carry out the optimisation process more quickly than PEST because it does not have the same overheads as PEST does. Furthermore, MODFLOW-2000 will sometimes require fewer optimisation iterations to lower the objective function to its final value than PEST requires. This is because default values for PEST control variables supplied by MF2PEST (particularly the values supplied for RLAMBDA1 and other variables which govern the calculation of the Marquardt lambda) are better suited to parameter estimation problems where difficulties may be encountered due to parameter correlation and insensitivity. When generating the PEST input dataset, MF2PEST "errs on the side of caution" in supplying default values for PEST control variables because parameter estimation problems are, in general, more likely to be difficult than easy.

Another point to keep in mind when comparing the results of a MODFLOW-2000 parameter estimation run with the results of a PEST run is that upper and lower parameter bounds are rigidly enforced by PEST whereas they are not enforced by MODFLOW-2000 at all. PEST will inform you through its screen output and its run

record file if any parameter is constrained in its movement by its upper or lower bound. Because no such limitation is incurred by a MODFLOW-2000 parameter, it is possible that MODFLOW-2000 will lower the objective function in a particular parameter estimation problem to a lower value than PEST does, especially where the "allowed parameter space" as far as PEST is concerned (ie. the space between parameter upper and lower bounds) is narrow.

Further differences between the objective functions calculated by PEST and MODFLOW-2000, and between parameters estimated by these two packages may be induced by operation of MODFLOW "drying/re-wetting" functionality. As is explained in Section 5, MF2KASP (the version of MODFLOW-2000 supplied with MF2PEST) provides the user with options for handling dry cell conditions additional to those provided by MODFLOW-2000. Use of one in particular of these MODFLOW enhancements in conjunction with PEST is mandatory; as is explained in Section 5, its use will result in slightly different objective function values and possibly slightly different parameter estimates.

## 2.8 Reducing Screen Clutter

When PEST runs a model (in this case the model is either of the two model batch files *model1.bat* or *model2.bat*), both PEST and the model share the same screen (unless WinPEST is used). Thus PEST's screen output scrolls upwards and out of site as MODFLOW output is written to the screen (and vice versa). You can prevent this from occurring by editing both of the model batch files written by MF2PEST in such a way as to direct MODFLOW output to a "nul" file. Figure 2 shows one such modified batch file.

```
@echo off
mf2kasp < pmodrun1.in > nul
```

**Figure 2. Re-directing MODFLOW output to a "nul" file.**

The first line of the batch file featured in Figure 2 suppresses the echoing to the screen of commands listed in the batch file as they are executed by the operating system. The "@" character in front of the "echo" command, suppresses echoing of the "echo" command to the screen. If both of the batch files used by PEST to run MODFLOW are altered in the manner described above, then only PEST-generated screen output will actually be written to the screen.

## 2.9 After A PEST Run

After PEST has finished execution, optimised parameter values will be found in both the PEST run record file and in the PEST parameter value file. The latter file has the same filename base as the PEST control file ("*pestrun*" if the default MF2PEST name is used), but carries the extension of "*.par*". Parameter sensitivities will be found in the sensitivity

file (extension of "*.sen*") while residuals and various functions of residuals will be found in the residuals file (extension of "*.res*").

After a PEST run, you may wish to load optimised parameter values into a MODFLOW-2000 input dataset. The latest MODFLOW run undertaken by PEST will not necessarily have been undertaken using the best set of parameters. In order to undertake a model run on the basis of optimised parameter values so that the MODFLOW-2000 output files generated by that run can be used for plotting results, inspecting MODFLOW-calculated statistics etc, program PAR2SEN can be used to transfer parameter values calculated by PEST to an existing MODFLOW-2000 *sensitivity* process input file. MODFLOW-2000 can then be run using optimised parameter values.

PAR2SEN is run using the command:-

```
par2sen parfile senfile
```

where *parfile* is the name of a PEST parameter value file and *senfile* is the name of a MODFLOW-2000 *sensitivity* process input file. Where a parameter name from the parameter value file matches a MODFLOW-2000 parameter name recorded in the *sensitivity* process input file, PAR2SEN replaces the parameter value found in the latter file by that found in the former file. *senfile* in the above command should be the *sensitivity* process input file cited in the MODFLOW-2000 name file that was read by MF2PEST when it generated the PEST input dataset. Recall that the name of this name file was provided to MF2PEST as the value of the MODNAMFILE variable illustrated in Figure 1; this is the same file as that written by the MODFLOW graphical user interface which generated the MODFLOW-2000 input dataset in the first place. You should place optimised parameter values into the MODFLOW-2000 *sensitivity* process file, rather than into either of the *sensitivity* process input files produced by PEST on the basis of template files written by MF2PEST, because MF2PEST alters these latter sensitivity files slightly from the original *sensitivity* process input file in the manner already described.

When running MODFLOW-2000 on the basis of these optimised parameters, there is no need to repeat the parameter estimation process. So you should edit the MODFLOW-2000 name file, placing a "#" (ie. comment character) in front of the "*pes*" string, thereby disabling the MODFLOW-2000 *parameter estimation* process. Alternatively, keep the *parameter estimation* process active, but alter the value of the MAX-ITER variable in the MODFLOW-2000 *parameter estimation* process input file to 0. MODFLOW-2000 will then simply calculate parameter statistics on the basis of parameter values supplied in the *sensitivity* process input file and, depending on the values of various control variables contained within the *parameter estimation* process input file, prepare input files for the post processing programs BEALE-2000 and YCINT-2000.

Upon commencement of execution of MODFLOW-2000 you will be prompted for its name file. Supply the name of the original MODFLOW name file (ie. the name file prepared by your graphical user interface). A plethora of MODFLOW-2000 outputs (for example sensitivities, heads/flows corresponding to field measurements, parameter

statistics, etc.) calculated on the basis of best-fit parameter values will then be at your disposal.

Note that when running MODFLOW-2000 in order to generate outputs corresponding to best-fit parameter values in this manner, you should run the modified MODFLOW-2000 supplied with MF2PEST (ie. MF2KASP) rather than the version of MODFLOW-2000 supplied by the USGS if you have used a non-zero value for the ASP input variable MINTHICK when calibrating the model (see Section 5 for further details). However if you do this you will not be able to obtain parameter sensitivities. Similarly MF2KASP, rather than the unaltered MODFLOW-2000, should be used when running MODFLOW to make predictions under these circumstances. In both cases an "ASP" entry must be added to the pertinent name file in order to instruct MF2KASP to read a file which informs it of the non-default MINTHICK value to use; see Section 5 for further details.

Note that, as explained in Section 5, if MF2KASP is run on the basis of a name file which does not include an "ASP" entry, then its outputs will be identical to those of the USGS version of MODFLOW-2000.

## 2.10 MODFLOW-2000 Sensitivity Process Input Files used by PEST

As has already been discussed, when PEST runs MODFLOW-2000 it uses a different name file from that written by your graphical user interface and supplied to MF2PEST as the variable MODNAMFILE in Figure 1. The original MODFLOW name file is not overwritten and is thus retained for later use. Similarly, when PEST writes *sensitivity* process input files for MODFLOW-2000 to use on particular model runs on the basis of the template files generated by MF2PEST, it does not use the same filename as that specified in the original MODFLOW name file (unless you specifically ask for this to happen – see later). Thus the original *sensitivity* process input file as written by your MODFLOW pre-processor remains intact (until you use PAR2SEN to replace parameter values found in that file by optimised parameter values as described in Section 2.8).

The *sensitivity* process input file written by PEST prior to each MODFLOW run is different from the *sensitivity* process input file supplied with the original MODFLOW dataset in the following ways:-

- Parameter values written to a PEST-generated *sensitivity* process input file are those which PEST would like MODFLOW-2000 to use on its next run. These vary from run to run.

- The ISENSU and ISENPU variables in the *sensitivity* process input file used by PEST are both set to 0; thus MODFLOW-2000 does not print or save sensitivity arrays.

- When a MODFLOW run is undertaken for the purpose of testing a parameter upgrade, the ISENALL *sensitivity* process input variable is set to -1. Thus

MODFLOW-2000 does not waste time calculating parameter sensitivities, as these are not required on a run of this type.

- Any parameters that were declared as being log-transformed for the purposes of sensitivity matrix calculation and parameter estimation in the original *sensitivity* process input file are declared as untransformed. Thus sensitivities produced by MODFLOW-2000 (where it is run by PEST for the purpose of derivatives calculation) are calculated with respect to native parameters rather than log-transformed parameters in accordance with PEST's requirements.

## 2.11 Another MF2PEST Option

A complete discussion of all the options available for MF2PEST usage will be provided in the next section. However this section will finish by describing just one of these options; this option provides the most useful extension of basic MF2PEST usage described so far. Figure 3 shows the pertinent MF2PEST input control file; this figure should be compared with Figure 1.

```
* modflow control
 MODNAMFILE d:\asps\test\tc1.nam
 MODFLOWCOM mf2kasp
 IMDERCALC 0
```

**Figure 3. The MF2PEST input control file of Figure 1 with the IMDERCALC variable added.**

A new control variable (named IMDERCALC) appears in the MF2PEST input control file of Figure 3. The default setting for IMDERCALC is 1, in which case PEST uses derivatives calculated by MODFLOW-2000 to fill its Jacobian matrix. However if IMDERCALC is set to 0, PEST calculates MODFLOW derivatives itself using finite differences. In this case, when PEST runs MODFLOW-2000, it never asks it to calculate derivatives, for it is only ever interested in those model outputs for which there are complementary field observations, ie. in the information contained in the "*._os*" file produced by MODFLOW-2000's *observation* process. Thus MF2PEST generates only one name file (with a default name of *pestrun1.nam*) and only one *sensitivity* process input file template (with a default name of *modsen1.tpl*). The MODFLOW *sensitivity* process input file expected by MODFLOW-2000 (ie. the file for which *modsen1.tpl* is the template) has a default name of *modsen1.sen*.

When derivatives are calculated by PEST rather than MODFLOW-2000, time required for completion of the optimisation process may be greater than when PEST can use derivatives calculated by MODFLOW-2000 itself. There are two reasons for this.

1. During each optimisation iteration PEST must run MODFLOW-2000 once for each adjustable parameter as it calculates the Jacobian matrix (twice if derivatives are calculated by central differences). When derivatives are calculated by MODFLOW-2000 itself, MODFLOW needs to be run only once during the Jacobian-calculation phase of each optimisation iteration. While MODFLOW-2000 does, indeed, have a lot more work to do when calculating sensitivities than when simply calculating heads, the time required for a MODFLOW-2000 sensitivity calculation run may still be significantly less than that required for the multiple MODFLOW runs needed to compute the same sensitivities by finite differences.

2. MODFLOW-2000 is able to calculate derivatives much more accurately internally than PEST can calculate them by finite differences on the basis of repeated model runs. Use of an accurate Jacobian matrix often allows the objective function to be minimised in fewer optimisation iterations than use of a lower accuracy Jacobian matrix such as that produced if finite differences are used to calculate derivatives.

*It is very important to note that if PEST is asked to calculated MODFLOW derivatives using finite differences, the MODFLOW solution convergence setting HCLOSE should be set smaller than normal. This minimises the loss of precision incurred through subtracting numbers of similar size through the finite-difference derivatives calculation process. A setting of $10^{-4}$ is often suitable. However if difficulties are encountered with MODFLOW solution convergence, it may have to be set higher.*

*It is also very important that attention be paid to PEST settings which govern the way in which finite-difference derivatives are calculated; see Sections 3.7 and 3.8 of this document for further details.*

In spite of the obvious disadvantages of using PEST to calculate MODFLOW derivatives, rather than allowing MODFLOW to calculate these derivatives itself, experience has shown that in many difficult cases (such as may be encountered where there is a high degree of cell drying and re-wetting), performance of the parameter estimation process is actually superior when IMDERCALC is set to 0 than when it is set to 1. Thus, in these circumstances, use of PEST to undertake the parameter estimation process is superior to the use of MODFLOW-2000. For reasons which are not too clear, PEST often has the ability to "break through" in difficult situations where, it seems, the slight loss of numerical precision incurred through the use of finite-difference based derivatives, in conjunction with PEST's strategic testing and selection of new Marquardt lambdas, actually appears to abet, rather than hinder, the parameter estimation process.

Note that if derivatives are calculated using finite differences, then Parallel PEST can be used in addition to the normal PEST.

## 2.12 A Note on Observation Names

It is not compulsory (though it is recommended), as far as MODFLOW-2000 is concerned, that every observation possess a different name. However for PEST it is compulsory. If MF2PEST discovers that there are observations of identical name in a MODFLOW-2000 input dataset, its course of action will depend on whether IMDERCALC is set to 1 or to 0. If it is set to 0, MF2PEST tries to re-name the observations so that each name is unique. If it can accomplish this, it informs the user of what it has done. If it cannot easily rename all observations such that each name is unique (while still retaining a semblance of the former names in the new set of observation names) it will cease execution with an appropriate error message. Note that observations are renamed only for the PEST input dataset; the original observation names are retained in the MODFLOW-2000 input dataset.

If IMDERCALC is set to 1, MF2PEST cannot rename observations, for MF2KASP's ability to generate external derivatives for the use of PEST requires that observations be named the same in the two packages. In this case, MF2PEST will cease execution with an appropriate error message.

*It is strongly suggested (GUI developers - take note), that each observation be assigned a different name in a MODFLOW-2000 input dataset, so that use can be made of PEST-ASP's capacity to use MODFLOW-2000-generated derivatives in the inversion process.*

# 3. Advanced MF2PEST Usage

## 3.1 MF2PEST Input Control File

As has already been discussed, the MF2PEST input file is subdivided into sections. In most of these sections, each line of the MF2PEST input control file should contain the name of an MF2PEST input control variable followed by the value of that variable. Except for filenames on a UNIX platform, all of the information contained in an MF2PEST input control file is case-insensitive.

The MF2PEST input control file will now be discussed section by section. This discussion will include the role of each MF2PEST input control variable, and will provide the default value for each such variable. As has already been discussed, you do not need to include any section in the MF2PEST input control file for which default values for all MF2PEST control variables contained within that section are acceptable, or for which the section is inapplicable to the current parameter estimation exercise; only the *modflow control* section is mandatory. Within any section of the MF2PEST input control file, it is only necessary to cite those variables whose values differ from MF2PEST defaults. Variables within a section can be listed in any order.

Any line within an MF2PEST input control file can be "commented out" through beginning it with the "#" character.

## 3.2 Modflow Control Section

Figure 4 shows an example of the *modflow control* section of an MF2PEST input control file. All variables which can appear in this section are listed in this figure.

```
* modflow control
MODNAMFILE study_site.nam
P1MODNAMFILE cal1.nam
P2MODNAMFILE cal2.nam
IMDERCALC 1
STMP1FILE sen1.tpl
STMP2FILE sen2.tpl
SEN1FILE sen1.sen
SEN2FILE sen2.sen
MODDERFILE paramderiv.dat
MODINSFILE flows.ins
MODFLOWCOM mf2kasp
MOD1FILE model1.bat
MOD2FILE model2.bat
MOD1RESFILE modres1.in
MOD2RESFILE modres2.in
NOSTOP 1
HDRYBOT 1
LIMOP 1
MINTHICK 0.0
ASPINFILE1 file1.asp
ASPINFILE2 file2.asp
```

**Figure 4. An example of the *modflow control* section of an MF2PEST input control file.**

Default values for all of the variables shown in Figure 4 are provided in Table 1.

| MF2PEST variable name | Default value |
|---|---|
| MODNAMFILE | no default value |
| P1MODNAMFILE | *pestrun1.nam* |
| P2MODNAMFILE | *pestrun2.nam* |
| IMDERCALC | 1 |
| STMP1FILE | *modsen1.tpl* |
| STMP2FILE | *modsen2.tpl* |
| SEN1FILE | *modsen1.sen* |
| SEN2FILE | *modsen2.sen* |
| MODDERFILE | *modflow.der* |
| MODINSFILE | *modobs.ins* |
| MOD1FILE | *mod1run.bat* |
| MOD2FILE | *mod2run.bat* |
| MOD1RESFILE | *pmodrun1.in* |
| MOD2RESFILE | *pmodrun2.in* |
| MODFLOWCOM | *mf2kasp* |
| NOSTOP | 1 |
| HDRYBOT | 1 |
| LIMOP | 0 |
| MINTHICK | 0.0 |
| ASPINFILE1 | *setting1.asp* |
| ASPINFILE2 | *setting2.asp* |

**Table 1. Default values for variables comprising the *modflow control* section of an MF2PEST input control file.**

The roles of variables MODNAMFILE (the existing MODFLOW name file),

MODFLOWCOM (the command required to run the PEST-compatible version of MODFLOW-2000) and IMDERCALC (a flag to indicate whether derivatives are to be calculated by MODFLOW-2000 or by PEST) have already been discussed. Recall that if IMDERCALC is set to 1, MF2KASP will calculate derivatives of observations with respect to adjustable parameters itself, whereas if IMDERCALC is set to 0 PEST will calculate these derivatives using finite differences.

P1MODNAMFILE and P2MODNAMFILE are the names of the two MODFLOW name files that MF2PEST writes for the use of MODFLOW-2000 when it is run by PEST. PEST directs MODFLOW-2000 to the first of these name files when it runs MODFLOW in order to calculate model outputs corresponding to field observations. It directs MODFLOW-2000 to the second of these name files when it runs MODFLOW-2000 in order to calculate derivatives. As has already been explained, these two name files cite different MODFLOW *sensitivity* process input files and different ASP input files. The ASP input file cited in the latter name file directs the modified MODFLOW-2000 supplied with MF2PEST (ie. MF2KASP) to record sensitivities to a special file for the use of PEST in the inversion process.

Parameter values for the use of MODFLOW-2000 on any particular model run are provided to it through the *sensitivity* process input file. As has already been discussed, two such files are provided, one for the use of MODFLOW-2000 when it is run to calculate heads and flows only, and the other for its use when it is run to calculate derivatives. Names for the template files corresponding to these two MODFLOW-2000 *sensitivity* process input files are supplied to MF2PEST through the variables STMP1FILE and STMP2FILE. Names for the corresponding MODFLOW-2000 *sensitivity* process input files (which PEST writes before each pertinent model run on the basis of these template files) are provided through the variables SEN1FILE and SEN2FILE.

The modified version of MODFLOW-2000 that must be used with PEST (ie. MF2KASP) writes its derivatives to a "derivatives file" in a format expected by PEST for a file of this type; see documentation to PEST version 5.0 or later for details. MF2KASP obtains the name of this file by reading the PEST control file pertinent to the current parameter estimation problem (ie. the PEST control file written by MF2PEST); the name of the derivatives file expected by PEST is found in the *derivatives command line* section of the PEST control file. The default name for this file is *modflow.der*; if this name is not suitable for any particular occasion, an alternative name can be provided through the MF2PEST input variable MODDERFILE.

MODFLOW outputs for which there are measured equivalents are written by the MODFLOW *observation* process to an output file with an extension of "._*os*". The filename base of this file is supplied to MODFLOW as the variable OUTNAM in the *observation* process input file. PEST reads model-generated measurement equivalents using an instruction set contained in a file whose name is supplied through the MF2PEST input variable MODINSFILE.

MF2KASP is run by PEST through a batch file. If fact two batch files are used, one to run the model simply for the purpose of obtaining head and flow outputs, the other to run the model for the purpose of calculating derivatives. The names of these respective batch files are the values of the variables MOD1FILE and MOD2FILE. As has already been discussed, each of these files contains a single line containing the command to run MF2KASP followed by a file containing the response to MF2KASP's prompt for the name of its name file. The files containing the response to this prompt for the two types of MODFLOW run are named *pmodrun1.in* and *pmodrun2.in* if the MF2PEST default settings are accepted. Alternative names can be provided through the variables MOD1RESFILE and MOD2RESFILE. The first of these files contains the name of the name file that is supplied to MODFLOW-2000 when it is run for the purpose of computing head/flow observation equivalents only. The second of these files contains the name of the name file which MODFLOW-2000 should use when it is run to calculate derivatives.

The roles of the final 6 variables appearing in Table 1 will be clearer after reading Section 5 of this document. As will be explained in that section, MF2KASP requires a little extra input data in addition to that of the normal MODFLOW-2000. At the very least, values for 4 variables should be supplied, the names of these variables being NOSTOP, HDRYBOT, LIMOP and MINTHICK. These variables are supplied to MF2KASP in a special "ASP input file". MF2PEST writes this file itself. If derivatives are to be calculated by MF2KASP, then two such files are required, one for the use of MF2KASP when it is run simply for the purpose of calculating model outputs, and the other for its use when it is run for the purpose of calculating derivatives; in the latter case, settings within the ASP input file instruct MF2KASP to activate its PEST interface. Default names for these files are listed in Table 1.

As is explained in Section 5, it is mandatory that HDRYBOT be set to 1 if MF2KASP is used in conjunction with PEST; this is the MF2PEST default value for this variable. Though it is not the default condition, it is advisable that LIMOP also be set to 1 in order to reduce unwanted file output when MF2KASP is run as part of a parameter estimation process. However a negative repercussion of setting LIMOP to 1 is that model outputs will not be available for user inspection unless one extra model run is carried out after the parameter estimation process has reached completion, with LIMOP set to 0 for this run. An extra model run will normally be required anyway, because it is rare that the last model run undertaken by PEST will have been carried out using optimised parameter values. Nevertheless, the default value for LIMOP has been set to 1 rather than 0 in case the user forgets to re-set it to 1 him/herself before conducting the final model run and then wonders where all of MODFLOW's file output has gone. However, it is strongly advised that a user with a good memory set LIMOP to 1 during the parameter estimation process to avoid the generation of megabytes of unwanted data over the course of a parameter estimation run.

As is further explained in Section 5, if MINTHICK is greater than 0.0 (as can be very useful when modelling, large, regional aquifers, especially in areas of high basement relief), the sensitivity calculations of MODFLOW-2000 are invalidated. Hence

IMDERDALC must be set to 0 so that PEST can calculate derivatives itself during the parameter estimation process. If this is not done, MF2PEST will terminate execution with an appropriate error message.

As is also explained in Section 5, an ASP input file may contain variables governing the operation of MODFLOW-2000's internal array generation functionality. These modifications can be extremely useful where parameterisation is undertaken on the basis of pilot points. *If the existing MODFLOW name file for the current case already cites an ASP input file, then any parameters pertaining to array generation and spatial interpolation within that existing file are transferred to the new ASP input files written by MF2PEST.* (Note, however, that MF2KASP may object to certain settings of these parameters if it is asked to calculate sensitivities, and hence if IMDERCALC is set to 1.)

## 3.3 PEST Control Section

The *pest control* section of the MF2PEST input control file provides the means whereby you can instruct MF2PEST to use values of your choice for variables cited in the *control data* section of the PEST control file instead of MF2PEST default values for these variables. Figure 5 shows an example of the *pest control* section of an MF2PEST input file. In this example, every variable that is useable in the *pest control* section of an MF2PEST input control file is cited; MF2PEST default values are shown for each such variable. Any PEST control variable for which the MF2PEST default value is suitable for your application can be omitted from this section if desired. In fact, if you are happy with all of the default values provided by MF2PEST for these variables, this section can be omitted altogether.

```
* pest control
RSTFLE restart
PESTMODE estimation
PRECIS single
DPOINT point
RLAMBDA1 10.0
RLAMFAC 2.0
PHIRATSUF 0.3
PHIREDLAM 0.03
NUMLAM 10
RELPARMAX 5.0
FACPARMAX 5.0
FACORIG 0.001
PHIREDSWH 0.1
DOAUI noaui
NOPTMAX 50
PHIREDSTP 0.001
NPHISTP 4
NPHINORED 4
RELPARSTP 0.001
NRELPAR 4
ICOV 1
ICOR 1
IEIG 1
JACFILE 1
MESSFILE 0
NUMCOM 1
```

**Figure 5. An example of the *pest control* section of an MF2PEST input control file. Values shown for variables are the actual default values used by MF2PEST.**

See the PEST manual for the role of each of the variables appearing in Figure 5, and for the range of values that are appropriate for each of them.

## 3.4 Predictive Analysis Section

Unless directed otherwise through the PESTMODE control variable, MF2PEST assumes that PEST will run in parameter estimation mode. Hence it does not write a *predictive analysis* section in the PEST control file which it generates. However if, through the PESTMODE variable, you inform MF2PEST that you wish PEST to run in predictive analysis mode, then it will, indeed, include a *predictive analysis* section in the PEST control file which it writes. It is important to note, however, that MF2PEST cannot supply default values for all of the predictive analysis control variables required in the PEST control file as the values of some of these variables are case-specific. Hence if you designate PESTMODE as *prediction* in the *pest control* section of an MF2PEST input file, you must include a *predictive analysis* section in this same file; furthermore, this

section must contain values for at least those variables for which MF2PEST cannot supply default values.

Figure 6 shows an example of the *predictive analysis* section of an MF2PEST input control file. Values for all variables except NPREDMAXMIN, PD0, PD1 and PD2 are MF2PEST default values. Because NPREDMAXMIN, PD0, PD1 and PD2 are case-specific, values for these variables must be supplied by the user. If they are not, MF2PEST will cease execution with an error message.

```
* predictive analysis
 NPREDMAXMIN 1
 PD0 55.0
 PD1 60.0
 PD2 100.0
 ABSPREDLAM 0.0
 RELPREDLAM 0.005
 INITSCHFAC 1.0
 MULSCHFAC 1.5
 NSEARCH 1
 ABSPREDSWH 0.0
 RELPREDSWH 0.05
 NPREDNORED 4
 ABSPREDSTP 0.0
 RELPREDSTP 0.005
 NPREDSTP 4
```

**Figure 6. An example of the *predictive analysis* section of an MF2PEST input control file.**

As is explained in the PEST manual, if PEST is run in predictive analysis mode, one (and only one) observation must belong to an observation group named *predict*. MF2PEST does not check whether this group is actually cited in the *modflow observations* section of the MF2PEST input control file (see below). If it is not, and if MF2PEST thus omits this group from the PEST control file which it generates, it can be added later if you wish. If you forget to do this, PEST (or PESTCHEK) will soon detect the omission.

## 3.5 Regularisation Section

Unless directed otherwise through the PESTMODE control variable, MF2PEST assumes that PEST will run in parameter estimation mode. Hence it does not include a *regularisation* section in the PEST control file which it generates. However if, through the PESTMODE variable cited in the *pest control* section of an MF2PEST input control file, you inform MF2PEST that you wish to run PEST in *regularisation* mode, then MF2PEST will, indeed, include a *regularisation* section in a PEST control file which it writes. It is important to note, however, that MF2PEST cannot supply default values for all of the variables cited in the *regularisation* section of the PEST control file, for some of these are case-specific. Hence if you designate PESTMODE as “regularisation”, you

must include a *regularisation* section in the MF2PEST input control file; furthermore this section must contain values for at least those variables for which MF2PEST cannot supply default values.

Figure 7 shows an example of the *regularisation* section of an MF2PEST input control file. Most of the variables are easily recognised from the *regularisation* section of the PEST control file; for all of these variables except PHIMLIM and PHIMACCEPT, the values shown in the figure are MF2PEST default values. Default values cannot be supplied for PHIMLIM and PHIMACCEPT because of their case-specific nature; see the PEST manual for details. Hence if PEST is asked to run in regularisation mode, and values for either of these two variables are absent from the *regularisation* section of an MF2PEST input control file, MF2PEST will cease execution with an appropriate error message.

```
* regularisation
PHIMLIM 25.7
PHIMACCEPT 28.0
FRACPHIM 0.0
WFINIT 0.1
WFMIN 1.0E-5
WFMAX 100.0
WFFAC 1.3
WFTOL 0.01
REGPRIORFILE regprior.txt
```

**Figure 7. An example of the *regularisation* section of an MF2PEST input control file.**

As is explained in the PEST manual, if PEST is run in regularisation mode some observations or prior information equations must belong to an observation group named *regul*. MF2PEST does not check that such a group has been cited in the *modflow observations* section of its input control file. If it is not, and if MF2PEST thus omits this group from the PEST control file which it generates, it can be added to that file later. If you forget to do this, PEST (or PESTCHEK) will soon detect the omission.

In many cases of practical interest, regularisation information will be supplied to the parameter estimation process through a series of prior information equations expressing the fact that the preferred differences between the values of neighbouring parameters is zero. (This is particularly useful where spatial parameterisation is undertaken through the use of pilot points.) Where the logs of parameters are being estimated through the parameter estimation process, then these differences must refer to the logs of parameter values rather than to the parameter values themselves. Unfortunately MODFLOW-2000 cannot accommodate prior information equations involving more than one log-transformed parameter. Thus if PEST is to be used in regularisation mode under these conditions, these prior information equations must be supplied to PEST "from outside". A number of mechanism are available by which this can be achieved. Prior information equations can be written to the PEST control file generated by MF2PEST directly by the

user. Or they can be added to this file by the utility program PPKREG supplied with the PEST Groundwater Data Utilities. Or they can be written to a "regularisation prior information file" (whose name is supplied following the REGPRIORFILE keyword – see Figure 7) and transferred to the MF2PEST-generated PEST control file by MF2PEST itself.

Figure 8 shows an example of a regularisation prior information file.

```
3
regprior1 1.0 * log(pp1) - 1.0 * log(pp2) = 0.0 1.0 regul
regprior2 1.0 * log(pp2) - 1.0 * log(pp3) = 0.0 1.0 regul
regprior3 1.0 * log(pp3) - 1.0 * log(pp2) = 0.0 1.0 regul
```

**Figure 8. An example of a regularisation prior information file.**

The first line of a regularisation prior information file must contain a single integer, this being the number of items of prior information to follow. Then must follow the prior information equations themselves. It is important to note that when MF2PEST transfers these items of prior information to the PEST control file, it does not check them or parse them. Therefore it is entirely the responsibility of the user to make sure that these are supplied with the correct syntax, that the number of equations cited at the top of the file agrees with the number of equations that are actually supplied, that prior information names are unique, that parameter names (and transformation states) are correct, etc. It is also important for the user to ensure that all items of prior information are assigned to the observation group *regul*.

## 3.6 Modflow Observations Section

MODFLOW-2000 observations can be of a number of different types, viz. head, drain, drain with return flow, river, general head boundary, stream, constant head boundary or advection. MF2PEST automatically assigns observations of these different types to different PEST observation groups, the names of these groups being *head*, *drain*, *drt*, *river*, *ghb*, *stream, const_head*, *adv* and *prior_info* respectively. Prior information is assigned to group *prior_info* or *prior_cov* depending on whether a covariance matrix was supplied or not.

In some instances you may wish to assign some MODFLOW observations to groups of a different name. For example, you may wish to monitor the contribution made to the objective function by measurements made in a certain group of observation bores, or by different segments of a stream. Alternatively, you may wish to assign a particular observation to the group "predict" as part of the preparation for a predictive analysis run, or you may wish to assign a suite of observations to the observation group "regul" so that PEST can run in regularisation mode.

Like every other section of the MF2PEST input control file, the *modflow observations* section must begin with a section header; in this case the header must be "modflow observations". However unlike the sections of the MF2PEST input control file that have

been discussed so far, each line within the *modflow observations* section must contain four entries. The first entry in each case must be the string *OBGNME*, ie. the name of the PEST variable describing the observation group name to which each observation is assigned. Figure 9 shows an example of the *modflow observations* section of an MF2PEST input control file.

```
* modflow observations
 OBGNME  riv?obs   river      reach_1
 OBGNME  bore_32   head       predict
 OBGNME  g*        ghb        flows
 OBGNME  b*d       prior_info pumptest
```

**Figure 9. An example of the *modflow observations* section of an MF2PEST input control file.**

The second entry on each line of the *modflow observations* section of an MF2PEST input control file should provide either the name of a MODFLOW observation (as defined in a MODFLOW-2000 *observation* input file for a particular package), or a text string from which one or a group of MODFLOW observation names can be inferred. The third entry must contain the name of a pre-defined observation group (see above), while the fourth entry must contain the name of a new observation group to which the selected observation(s) will be assigned.

To illustrate the role of variables appearing in the *modflow observations* section of an MF2PEST input control file, consider the example shown in Figure 9. In this example the observation named *bore_32* belonging to the existing observation group *head* will be assigned to an observation group named *predict* in the PEST control file generated by MF2PEST. Observations of type *river* with names such as *riv1obs, riv2obs* etc, will be assigned to an observation group named *reach_1*. All general head boundary observations with names beginning with "g" will be assigned to an observation group named *flows*. MODFLOW-2000 prior information equations whose labels begin with "b" and end with "d" will be assigned to an observation group named *pumptest*.

The protocol for text substitution for items appearing in the second column of Figure 9 is that the "?" symbol represents only a single character in the observation name which it represents, while the "*" symbol represents multiple characters. This is in accordance with common usage for these symbols in most computing environments.

Note that observations to which a covariance matrix (rather than observation weights) is assigned in the MODFLOW-2000 input dataset should **not** be assigned to an observation group in the manner described above; the default MODFLOW-2000 observation group name should be used for such observations. Recall that a covariance matrix can be supplied for all MODFLOW-2000 observation types except for head observations.

As is documented in the MODFLOW-2000 manual, two types of prior information items may reside in the MODFLOW-2000 *parameter estimation* process input file. For the first

type a covariance matrix is supplied; however these items of prior information are not given names in the *parameter estimation* process input file. In contrast, names are provided for each incidence of the second type of prior information, but weights are used instead of a covariance matrix for specification of observation uncertainty. In the former case MF2PEST assigns default prior information names in the format *pr_cov_n*, where *n* is the number of the prior information item. Hence a string such as "pr_cov_*" can be used in the *modflow observations* section of an MF2PEST input control file to assign such observations to a different observation group (for example to the group "regul" if such items of prior information comprise the "regularisation observations" required by PEST if it is working in regularisation mode).

## 3.7 Modflow Parameters Section

The *modflow parameters* section of an MF2PEST input control file is also a little different from most of the other sections of this file. While, like the other sections, each line must possess two entries, the first being the name of an MF2PEST (and PEST) variable and the second being its value, the *modflow parameters* section is subdivided into segments, each segment pertaining to one particular parameter named in the MODFLOW-2000 *sensitivity* process input file. Figure 10 shows an example of a *modflow parameters* section of an MF2PEST input control file. As is illustrated in this figure, each segment must begin with a parameter name (assigned to the PARNME variable) and end with an *END* keyword followed by the name of the same parameter.

```
* modflow parameters
parnme hk_1
  partrans tied
  pargp hcond
  partiedto hk_2
end hk_1
parnme hk_3
  partrans log
  parval1  1.24e-4
  parubnd  1e-2
  parlbnd  1e-5
  parchglim factor
  pargp hcond
  scale 1.0
  offset 0.0
end hk_3
```

**Figure 10. An example of the *modflow parameters* section of an MF2PEST input control file.**

Within each parameter segment of the *modflow parameters* section you can supply a value for any variable appearing in the *parameter data* section of a PEST control file. In doing this you can override the values for these variables that MF2PEST reads from the

MODFLOW-2000 *sensitivity* process input file, or the MF2PEST default values for those variables whose values are not obtained from this source. For example, you can assign a different initial value (PEST variable PARVAL1) as well as different upper and lower bounds (PEST variables PARLBND and PARUBND) to a parameter from those appearing in the *sensitivity* process input file. Also, if a parameter is designated as log-transformed in the *sensitivity* process input file, it can be designated as undergoing no transformation (ie. PARTRANS value of *none*) in the MF2PEST input control file. Similarly, the change limit type (PEST variable PARCHGLIM) can be supplied by the user, overriding MF2PEST's default assignment of this variable which depends on parameter upper and lower bounds and transformation status as described above.

A parameter can be designated as tied to another parameter. As is described in the PEST manual, this is achieved by supplying it with a PARTRANS value of "tied". If this is done, you must include a value for the PARTIEDTO variable in the same segment, this being the name of the parent parameter to which the parameter is tied. (There may, or may not, be a segment pertaining to this parameter in the *modflow parameters* section of the MF2PEST input control file.)

A parameter is assigned to a parameter group through the PARGP variable (named after the corresponding PEST variable). Unless a parameter is specifically assigned to a group in this manner, it is assigned to a default parameter group named *general* (for MF2PEST cannot determine from the information contained in the *sensitivity* process input file what type of property or characteristic of the system a particular parameter represents). However parameters groups are only important in the assignment of variables for finite-difference derivatives calculation. If derivatives are calculated by MODFLOW-2000 rather than by PEST (ie. the MF2PEST IMDERCALC variable is set to 1), then the group to which a parameter is assigned has no relevance because the variables governing derivatives calculation are never used by PEST. However if derivatives are calculated by PEST using finite differences (ie. the MF2PEST IMDERCALC variable is set to 0), then the values of variables assigned to each parameter group (all of which govern the way in which derivatives are calculated for members of that group), may be of great importance.

You can assign parameters to any one of 16 pre-defined parameter groups; alternatively you can assign a parameter to a parameter group which you define yourself in the *parameter groups* section of the MF2PEST input control file (see below). Pre-defined parameter groups, and the aquifer properties that they represent are listed in Table 2.

| Parameter Group Name | Parameter Type |
|---|---|
| *kh* | horizontal hydraulic conductivity |
| *hani* | horizontal anisotropy |
| *vk* | vertical hydraulic conductivity |
| *vani* | vertical anisotropy |
| *ss* | storage capacity |
| *sy* | specific yield |
| *vkcb* | vertical hydraulic conductivity of confining bed below a layer |
| *hfb* | hydraulic characteristic of hydraulic flow barrier |
| *riv* | riverbed conductance |
| *rch* | recharge rate |
| *q* | well pumping rate |
| *drn* | drain conductance |
| *drt* | conductance of drain with return flow |
| *evt* | maximum rate of evapotranspiration (evt package) |
| *ets* | maximum rate of evapotranspiration (ets package) |
| *ghb* | conductance of general head boundary |
| *chd* | boundary head |
| *str* | streambed conductance |

**Table 2. MF2PEST pre-defined parameter groups.**

MF2PEST assigns default values to variables governing derivatives calculation for

members of each of the groups listed in Table 2. These values cannot be altered. If you wish that a parameter be allocated to a group for which values for variables governing derivatives calculation can be assigned in the MF2PEST input control file, then assign that parameter to a group which you define yourself in the *parameter groups* section of the MF2PEST input control file (see below).

Values for variables pertaining to most of the default parameter groups listed in Table 2 are recorded in Table 3.

| Variable Name | Value |
|---|---|
| *INCTYP* | "relative" |
| *DERINC* | 0.01 |
| *DERINCLB* | 0.00 |
| *FORCEN* | "switch" |
| *DERINMUL* | 2.0 |
| *DERMTHD* | "outside_pts" |

**Table 3. Values of the derivative control variables pertaining to most of the groups listed in Table 2.**

According to Table 3, derivatives for parameters belonging to the pertinent parameter groups are calculated using an increment equal to 1% of the current parameter value. When the relative improvement in the objective function between optimisation iterations is less than the value of the PEST control variable PHIREDSWH, a switch is automatically made to three-point derivatives calculation and the increment is doubled. When three-point derivatives calculation is employed, linear interpolation between the outside points is used to calculate the derivative at the current parameter value.

Exceptions to the default parameter values listed in Table 3 for those groups which are listed in Table 2 are as follows:-

- *INCTYP* for parameter groups *evt*, *ets* and *rch* is set to "rel_to_max".
- *INCTYP* for parameter group *chd* is set to "absolute".
- The absolute derivative increment (ie. DERINC) for parameter group *chd* is set to 0.01.

Before running PEST you should make sure that derivative control variables written to the PEST control file by MF2PEST are suitable for your particular parameter estimation

problem. Particular attention should be given to the variables pertaining to groups *rch*, *q evt*, *ets* and *chd*. However, as has already been discussed, there is no need to worry about any of these derivative control variables if the MF2PEST control variable IMDERCALC is set to 1, because in this case derivative control variables are not needed due to the fact that derivatives will be calculated by the model.

## 3.8 Parameter Groups Section

The structure of the *parameter groups* section of the MF2PEST input control file is very similar to that of the *modflow parameters* section of this file. An example is shown in Figure 11.

```
PARGPNME group_1
  INCTYP absolute
  DERINC 0.01
  DERINCLB 0.0
  FORCEN switch
  DERINCMUL 2.0
  DERMTHD parabolic
END group_1
PARGPNME group_2
  DERINCLB 3.00
END group_2
```

**Figure 11. Part of the *parameter groups* section of an MF2PEST input control file.**

Like the *modflow parameters* section, the *parameter groups* section is divided into segments, each segment pertaining to one particular parameter group. The segment must begin with the *PARGPNME* keyword followed by the name of the parameter group, and must finish with the *END* keyword followed by the name of the same parameter group. You can choose any name for a parameter group, as long as that name is 12 characters or less in length and is not the string *general* or any of the names listed in Table 2.

Within each segment, values are supplied for PEST parameter group variables (these are the variables which govern derivatives calculation). On each line of this segment the name of a variable is listed, followed by the value of that variable. Variables can be supplied in any order. It is not necessary that, for any particular parameter group, you supply values for all of the PEST variables which govern derivatives calculation. Default values are assigned to variables which are omitted, the default values being those listed in Table 3.

## 3.9 General Section

The *general* section of an MF2PEST input control file can contain any or all of the variables which are listed in Figure 12. Default values for these variables are also shown in Figure 12.

```
* general
PESTCTLFILE pestrun.pst
DRNCOVFILE drain.cov
DRTCOVFILE drt.cov
RIVCOVFILE riv.cov
GHBCOVFILE ghb.cov
STRCOVFILE str.cov
CHDCOVFILE chd.cov
ADVCOVFILE adv.cov
PICOVFILE pi.cov
```

**Figure 12. An example of the *general* section of an MF2PEST input control file.**

PESTCTLFILE is the name of the PEST control file which MF2PEST must write. If a value for this variable is not supplied, the default name of *pestrun.pst* is used.

The other variables belonging to the *general* section of the MF2PEST input control file are the names of the files to which covariance matrices will be written if these are provided in the MODFLOW-2000 input dataset. Recall that in preparation of an input dataset for PEST, covariance matrices are written to separate files; the names of these files are then cited in the *observation groups* section of the PEST control file.

## 3.10 Automatic User Intervention Section

Versions of PEST from 6.0 onwards include functionality for "automatic user intervention" (AUI). This is an extremely powerful method for mitigating the deleterious effects of parameter insensitivity on the inversion process. When AUI is activated, insensitive parameters are selectively held at their current values at various stages of the optimisation process. The result is a better conditioned "normal matrix" (PEST must invert the normal matrix when calculating a new parameter upgrade vector), and a lessening of the dampening effects of the limits that PEST is forced to impose on the length of the parameter upgrade vector to prevent this length from grossly exceeding the range of the linearity assumption upon which its calculation is based.

As is discussed in the PEST manual, AUI is activated through setting the DOAUI variable in the *control data* section of the PEST control file to "aui". Implementation of AUI requires that values be assigned to 10 variables which control the operation of this process. Optionally, an *automatic user intervention* section can be present within a PEST control file containing user-assigned values for these variables. If DOUI is set to "aui" and this section is not present, PEST will supply default values for all of these AUI variables itself.

If AUI is required in a PEST run based on a control file written by MF2PEST, this can be activated by assigning the string "aui" to the DOAUI variable in the *pest control* section of the MF2PEST input control file. Optionally, an *automatic user intervention*

section can also be present within an MF2PEST input control file for the assignment of values to some or all AUI variables. Figure 13 illustrates such a section. Default values are assigned to variables which do not appear in this section (or to all AUI variables if the section is absent altogether).

```
* automatic user intervention
MAXAUI 5
AUISTARTOPT 1
NOAUIPHIRAT 0.9
AUIRESTITN 0
AUISENSRAT 80.0
AUIHOLDMAXCHG 0
AUINUMFREE 2
AUIPHIRATSUF 0.8
AUIPHIRATACCEPT 0.96
NAUINOACCEPT 4
```

**Figure 13. An example of the *automatic user intervention* section of an MF2PEST input control file.**

## 3.11 Additional Parameters Section

Extra parameters, additional to those defined in the MODFLOW SEN file, can be included in the parameter estimation process. It should be noted however that, as presently coded, IMDERCALC should be set to zero for this to occur; thus PEST will calculate all derivatives (including those pertaining to MODFLOW parameters) by finite differences. Figure 14 shows an example of the *additional parameters* section of an MF2PEST input control file.

```
* additional parameters
PARAMDATFILE    param.dat
PARAMGROUPFILE  pargroup.dat
ADDCOMFILE      inter.dat
ADDTPLFILE      extra.tpl
ADDMODINFILE    extra.in
ADDTPLFILE      extra1.tpl
ADDMODINFILE    extra1.in
```

**Figure 14. An example of the *additional parameters* section of an MF2PEST input control file.**

All of the variables in the *additional parameters* section of the MF2PEST input control file are filenames; all of these files (except perhaps for ADDMODINFILE) should be prepared by the user prior to carrying out the parameter estimation process.

PARAMDATFILE is the name of a "parameter data file" Such a file is illustrated in Figure 15.

```
ro1   fixed     factor  0.5   .1  10     ro    1.0   0.0
ro2   log       factor  5.0   .1  10     ro    1.0   0.0
ro3   tied_ro1  factor  0.5   .1  10     ro    1.0   0.0
h1    none      factor  2.0   .05 100    h     1.0   0.0
h2    none      factor  5.0   .05 100    h     1.0   0.0
```

**Figure 15. A parameter data file.**

For the most part, a parameter data file emulates the "parameter data" section of a PEST control file, containing the same variables in the same order. However, note the following.

1. There is no need to supply a value for the DERCOM variable (the command line number for derivatives calculation - the 10th variable on each line of the "parameter data" section of a PEST control file). MF2PEST will always provide a default value of 1 for this variable when it writes a PEST control file.

2. All parameters cited in "additional template files" (which are provided following ADDTPLFILE keywords – see below) need to be cited in the parameter data file. Conversely, any parameter cited in the parameter data file must be cited in at least one additional template file. MF2PEST will not detect non-adherence to this rule. However if this rule is not obeyed, the PEST input dataset written by MF2PEST will not be consistent; this inconsistency will be detectable by PESTCHEK.

3. If a parameter is tied to another parameter, the name of the parent parameter must be attached to the "tied" string following an underscore, as illustrated in the above example.

4. If a parameter is assigned to a particular parameter group, that group must be cited in a "parameter group file" – see below.

5. Parameter names cited in a parameter data file must not coincide with MODFLOW parameters cited in the MODFLOW SEN file.

The name of a parameter group file must follow the PARAMGROUPFILE keyword. An example of a parameter group file is provided in Figure 16. The contents of a parameter group file emulate those of the "parameter groups" section of the PEST control file. See the PEST manual for details. Note that parameter group names provided in this file must be different from group names provided in the *parameter groups* section of the MF2PEST control file, and from the names of pre-defined parameters groups listed in Table 2.

```
ro  relative  0.01  0.00    switch 1.5 parabolic
h   relative  0.01  1.0e-4  switch 2.0 parabolic
```

**Figure 16. A parameter group file.**

The contents of the parameter data file and the parameter group file are transferred directly to the PEST control file written by MF2PEST; it is the user's responsibility to ensure consistency between these files.

As mentioned above, every parameter listed in the parameter data file must be cited in one or more "additional template files". The names of these files are supplied following ADDTPLFILE keywords in the *additional parameters* section of the MF2PEST control file. A minimum of one, and a maximum of ten, template files can be provided. For every template file a corresponding model input file must be provided; each such file is supplied following an ADDMODINFILE keyword. Template filenames are linked to model input filenames by order of appearance; to avoid confusion it is best to supply them paired as in Figure 14 above.

When additional parameters are added to the inversion process, it will often be necessary for the model batch file run by PEST to include extra commands. These must be supplied in a batch file whose name is provided following the ADDCOMLINE keyword. The contents of this file are written directly to the model batch file cited in the PEST control file written by MF2PEST, immediately before the command to run MF2K.

## 3.12 Modifying the PEST Control File

In many cases, preparation for a PEST run designed to calibrate a MODFLOW model will involve nothing more than running MF2PEST in order to convert a MODFLOW-2000 input dataset into a PEST input dataset. In other cases it may be necessary to modify the PEST control file produced by MF2PEST before PEST is run (see below). However in either case you should never forget to run PESTCHEK in order to check the entire PEST input dataset before running PEST, whether or not any alterations were made to any of the files generated by MF2PEST.

The fact that a user can modify the behaviour of MF2PEST through the provision of preferred values for most PEST control variables through the MF2PEST input control file will obviate the necessity to edit the PEST control file directly on many occasions. However there will be other occasions when this cannot be avoided, the most likely occasions being those where a model other than MODFLOW is run together with MODFLOW as part of a composite model calibrated by PEST. The other model may be run prior to MODFLOW (for example a recharge model) or after it (for example a transport model such as MT3DMS). Extra parameters may require adjustment in the calibration of the composite model, and/or extra observations may constitute the dataset upon which the calibration process is based; these parameters and/or observations will need to be added to the PEST control file generated by MF2PEST. In either case,

through appropriate settings of various PEST control variables, it will be possible to obtain derivatives of at least some model outputs with respect to at least some adjustable parameters directly from MODFLOW-2000 itself, while others will require the entire composite model be run with pertinent parameters incremented and/or decremented so that derivatives can be calculated using finite differences in the normal manner. The user should bear in mind, however, that if the opportunity exists to obtain at least some derivatives from MODFLOW-2000, that opportunity should be grasped, for such derivatives will be calculated more quickly and more accurately by MODFLOW than they will be by PEST on the basis of finite differences (provided, of course, that MODFLOW is well-behaved and is not beset by drying/re-wetting problems).

Another instance where a composite model may need to be run by PEST is where parameter regularisation is undertaken. Here a submodel may be used to calculate the model-equivalents to a number of "observations" which quantify, for example, the degree of spatial variability that neighbouring parameter values are allowed to possess. Such "observations" will be assigned to the observation group "regul" and that observation group added to existing observation groups in the PEST control file generated by MF2PEST.

# 4. Program PAR2SEN

## 4.1 Role of PAR2SEN

After a PEST run, optimised parameter values will be listed in a "parameter value file" written by PEST. This file has the same filename base as that of the PEST control file, but possesses an extension of "*.par*". Normally it will be necessary to transfer these to a MODFLOW-2000 *sensitivity* process input file before running MODFLOW-2000 in order to calculate model outputs, sensitivities and various other statistics based on optimised parameter values. This task can be accomplished using program PAR2SEN. (Don't forget to disable the MODFLOW-2000 *parameter estimation* process by "commenting out" the pertinent line from the MODFLOW-2000 name file before you run MODFLOW-2000.)

## 4.2 Running PAR2SEN

PAR2SEN is run using the command:-

```
par2sen parfile senfile
```

where *parfile* is the name of a PEST parameter value file and *senfile* is the name of a MODFLOW-2000 *sensitivity* process input file. If PAR2SEN is run after a PEST run whose input files were created using MF2PEST on the basis of default filenames supplied by MF2PEST for PEST input files, then the parameter value file will be named *pestrun.par*.

PAR2SEN reads both the parameter value file and the MODFLOW-2000 *sensitivity* process input file cited in its command line. For every parameter listed in the MODFLOW-2000 *sensitivity* process input file, PAR2SEN looks for a parameter of the same name in the PEST parameter value file. If it does not find a parameter of the same name, it ceases execution with an error message. Otherwise, it reads a new value for that parameter from the parameter value file. Once it has found a new value for each of the MODFLOW-2000 parameters featured in the *sensitivity* process input file, it overwrites this file with a new one in which the new parameter values are listed. However, before overwriting it, PAR2SEN copies the original *sensitivity* process input file to a file of the same name, but with the suffix of ".kp" attached.

Note that while it is important that every parameter listed in the MODFLOW-2000 *sensitivity* process input file also be featured in the PEST parameter value file, the reverse is not necessary. For example a parameter value file can have more parameters than those listed in the *sensitivity* process input file if MODFLOW-2000 was run as part of a composite model calibrated by PEST, and parameters for some of the other models comprising this composite model were adjusted at the same time as the MODFLOW parameters.

# 5. MODFLOW-ASP

## 5.1 Introduction

As has already been discussed in Section 1, a number of minor modifications have been made to MODFLOW-2000 in order to allow it to work with PEST. These alterations mainly endow MODFLOW-2000 with the capacity to provide PEST with derivatives information, enhance its drying/re-wetting functionality, and enhance its ability to undertake parameterisation based on pilot points.

It will be recalled from the PEST (version 5.0 or later) manual that if PEST can obtain derivatives directly from a model, it does not need to calculate them by finite differences. Where the model can calculate derivatives quickly and accurately, it is far better to use these derivatives than finite difference derivatives. PEST reads model-calculated derivatives from a "derivatives file" written by the model; this file must adhere to the formatting conventions expected by PEST.

MODFLOW-2000 has been modified in order to generate a file of the type required by PEST in which the derivatives of all observations with respect to all parameters are recorded; the modified MODFLOW is named MF2KASP. As is described below, in addition to the inclusion of functionality for recording derivatives, certain other changes have been made to MODFLOW-2000 that affect the way in which MF2KASP handles dry cells, the amount of file output which it generates (most of which is unnecessary when being run repeatedly under the control of PEST as part of a parameter estimation process), and the manner in which certain arrays are calculated from pertinent parameter values (this being particularly useful where a pilot points scheme is used for spatial property definition - see Doherty, 2001). Parameters that govern the implementation of this added functionality are supplied through a special "ASP input file". However this file is only required if the "ASP process" is activated; if this process is not activated, then performance of MF2KASP is identical to that of the normal MODFLOW-2000. Activation of the ASP process is achieved through adding an appropriate "ASP" entry to the MODFLOW name file in a similar manner to which other MODFLOW-2000 processes are activated/de-activated through the existence or otherwise of pertinent entries in the name file. As part of its role in preparing for a PEST run, MF2PEST includes this entry in the name files which it generates, and writes the ASP input files required by MF2KASP (a different file is required depending on whether MF2KASP is run for the purpose of calculating derivatives, or whether it is run simply for the purpose of calculating model outputs).

Because alterations have been made to the way in which MODFLOW-2000 handles drying/re-wetting conditions, heads and flows calculated by MF2KASP may be slightly different (certainly no worse, and often better) from those calculated by the USGS version of MODFLOW-2000. Therefore, a parameter set that is optimum (in terms of model calibration) for the modified MODFLOW-2000 (ie. MF2KASP) may be a little

different from that which is optimum for the native MODFLOW-2000. Hence if you undertake the calibration process with altered drying/re-wetting settings in the modified MODFLOW (in particular, if you use a non-zero value for MINTHICK - see below), you should continue to use MF2KASP with the same MINTHICK setting when you run MODFLOW to make predictions. Alternatively, once the model has been calibrated, run both the native and modified MODFLOW using optimised parameters in order to compare results produced by the two versions. On most occasions differences will not be major; in that case you can use either one to make predictions.

## 5.2 MODFLOW Name File

As is explained in the documentation to MODFLOW-2000, MODFLOW is made aware of the processes and packages that are operative on any particular run through the entries in its name file. The first entry on each line of a MODFLOW name file contains the code for a file type, most of these file types being linked to a particular process or package.

Modifications made to MODFLOW-2000 for use with PEST are such that it will now recognise an extra code and file type in its name file. The code (first entry in the pertinent line) is "*asp*". Subsequent entries on the same line of the name file should be the unit number associated with that file, and the name of an existing PEST control file. Figure 17 shows such an entry in a MODFLOW-2000 name file. (Note that MF2KASP selects an appropriate unit number itself once execution has commenced; hence any integer in a MODFLOW name file will serve as a suitable ASP unit number for the purpose of activating this process.)

```
asp      55   input.asp
```

**Figure 17. The line in a MODFLOW name file that activates the ASP process.**

The contents and formatting of an ASP input file will be described below. As was mentioned above, various alterations have been made to MODFLOW-2000 to facilitate its use with PEST. Some or all of these new aspects of MODFLOW functionality can be activated through pertinent settings in the ASP process input file. Some of these items are essential for use of MODFLOW-2000 with PEST; others can be used or unused, depending on the demands of a particular problem. Some of them are useful whether or not MODFLOW-2000 is being run with PEST. Hence there will be some occasions when it is useful to activate various aspects of ASP functionality even if parameter estimation is being undertaken by the MODFLOW-2000 *parameter estimation* process rather than by PEST, or even if MF2KASP is being run simply for the purpose of calculating model outputs, independently of the parameter estimation process.

One of the most important items of functionality added to MODFLOW-2000 to form MF2KASP is its "PEST Interface" - the ability of MODFLOW-2000 to record derivatives for the use of PEST. This is now discussed in more detail.

## 5.3 What the MODFLOW-PEST Interface Does

If the MODFLOW-2000 PEST interface is activated through an appropriate setting in the ASP input file then, along with its usual input dataset, MF2KASP reads the PEST control file for the current case, the name of this file also being provided in the ASP input file. On reading this file it ensures that the *JACFILE* PEST control variable is set to 1; recall from the documentation to PEST version 5.0 or later that this variable informs PEST that it can expect model-generated derivatives to be found in a special derivatives file when the model is run specifically for the purpose of derivatives calculation. MF2KASP then reads the names of all parameters and observations cited in the PEST control file, as well as the name of the derivatives file expected by PEST; this name is provided in the *derivatives command line* section of the PEST control file.

As is discussed above, there will be many instances where MODFLOW is run in conjunction with another model as part of a composite model during the calibration process. Hence it is possible that there will be more parameters cited in the PEST control file than internal MODFLOW-2000 parameters that are designated as adjustable in its *sensitivity* process input file. However where a PEST parameter is, indeed, a MODFLOW-2000 parameter, it is essential that its name be the same in both the PEST and MODFLOW input datasets. (This will happen automatically if the PEST control file was written by MF2PEST.)

When PEST runs the modified MODFLOW-2000 with both the *sensitivity* and *PEST interface* processes active (which it does once per optimisation iteration when MODFLOW is run for the purpose of derivatives calculation), PEST interface code added to MODFLOW-2000 writes the derivatives file expected by PEST. In doing this it ensures that the arrangement of parameters and observations in this file is the same as that expected by PEST, and that where there are more observations and/or parameters cited in the PEST control file than are cited in the MODFLOW-2000 input dataset, that respective rows and/or columns of the derivatives matrix are filled with the appropriate "nul" value. See the PEST documentation for further details.

It is important to note that when writing the derivatives file, the MODFLOW-2000 PEST interface has been coded in such a way that it writes MODFLOW-2000 sensitivities "as it finds them". If a parameter is log-transformed, then the sensitivity of observations to this parameter will reflect its log-transformed status. If such sensitivities are then supplied to PEST it will have great difficulty in undertaking the parameter estimation process, because PEST expects native derivatives. Hence when MF2PEST prepares the template of the *sensitivity* process input file to be read by MODFLOW-2000 prior to a run in which it calculates derivatives for PEST, it ensures that no MODFLOW parameters are designated as log-transformed in this *sensitivity* process input file, even if they are log-transformed in the original MODFLOW-2000 dataset. In this way it is ensured that derivatives written by MODFLOW-2000 for the use of PEST are, indeed, calculated with respect to native parameters and not with respect to log-transformed parameters.

## 5.4 Drying and Re-Wetting

When the water level in a cell falls below the bottom of that cell, the normal USGS version of MODFLOW declares that cell as inactive. If MODFLOW's re-wetting functionality is active, the cell can later be re-activated either during the current heads solution process or later in the simulation. The de-activation and subsequent re-activation of cells in this manner is often disastrous for the parameter estimation process. Reasons for this are as follows:-

1. The de-activation/re-activation procedure normally involves the use of "re-wetting thresholds". This can result in a non-unique solution for cell heads. At the very least it will result in discontinuities in the relationships between MODFLOW outputs and the values of parameters adjusted through the parameter estimation process. The integrity of the Jacobian matrix will suffer as a result. Hence, whether calibration is undertaken by PEST or by the *parameter estimation* process of MODFLOW-2000, performance of this process is likely to be seriously degraded.

2. Even where cell de-activation/re-activation takes place at locations far removed from observation bores, the MODFLOW solution convergence criterion (normally named HCLOSE for the various MODFLOW solution packages) may need to be set quite high to prevent solution non-convergence. This can further detract from the accuracy of derivatives calculations if derivatives are calculated by finite differences using PEST, ie. if the MF2PEST IMDERCALC input variable is set to 0. (Recall however from a previous discussion in this document, that use of finite-difference-calculated derivatives in preference to MODFLOW-calculated derivatives can often allow the parameter estimation process to proceed even in difficult circumstances; nevertheless the process will not be optimum.)

3. If an observation bore falls within a cell that is declared as dry, or falls so close to such a cell that the spatial interpolation process by which borehole heads are calculated from MODFLOW heads is adversely affected, then the role of that bore in the parameter estimation process must be altered. If the bore is located within a dry cell, the USGS version of MODFLOW-2000 removes the bore from the parameter estimation process altogether (which can have an adverse effect on the parameter estimation process by artificially lowering the objective function). However if the bore is located in a cell which borders a dry cell but which is not itself dry, the spatial interpolation scheme is adjusted to exclude the dry cell from the interpolation process; the bore's status as part of the observation dataset upon which the calibration process is based is thus preserved. However variations such as this in the spatial interpolation process introduce discontinuities into the relationship between parameters and heads.

4. An undesirable situation can arise where head-dependent boundary conditions such as rivers, drains and general head boundaries are situated in upper layer cells, and flow from those boundaries is insufficient to maintain water levels in the respective cells above the bases of those cells. In this case, when MODFLOW declares the

respective cells as dry, the pertinent boundary conditions are lost from the system. This can alter the nature of the groundwater system irrevocably, resulting in a high degree of nonuniqueness in MODFLOW-calculated heads.

5. Use of flows from head-dependent boundaries as part of the observation dataset upon which the calibration process is based can often be extremely problematical. If a head-dependent boundary dries out because cells containing the boundary cannot maintain their wet status, the flow from that boundary, as calculated by MODFLOW, is zero. This can lead to a false representation of true field conditions. For example, flow from a river does not abruptly cease as the water table drops below a level designated as the bottom of the cell holding the river boundary condition. The discontinuous nature of this MODFLOW-calculated flow as the water table drops below the cell bottom may present serious numerical difficulties for nonlinear parameter estimation software which is working in conjunction with MODFLOW to undertake model calibration. MODFLOW-2000 attempts to overcome this problem by removing the observation from the calibration dataset when the flow becomes zero as a result of this effect. However this option is not available when using PEST to calibrate MODFLOW, for observations cannot be brought in and out of the parameter estimation process in this way unless special communication pathways were set up between PEST and MODFLOW to inform PEST of the status of various observations. So such observations must remain within the calibration dataset, and the discontinuous nature of corresponding model-generated flows must simply be tolerated.

If there is one rule that ought to be followed when calibrating MODFLOW using either the MODFLOW-2000 *parameter estimation* process, or using PEST-ASP, it is this:- *avoid drying and re-wetting like the plague.*

## 5.5 MF2KASP Settings

### *5.5.1 The ASP Input File*

Operation of functionality added to MODFLOW-2000 to form MF2KASP is governed by settings contained in an "ASP input file". The name of this file is provided in the MODFLOW-2000 name file for the current case. The specifications of this file are provided in Figure 18. Note that on most occasions of MF2KASP usage, the ASP input file is written automatically by the MODFLOW-2000 to PEST translator, MF2PEST, and hence does not need to be prepared by the user.

```
IPESTINT   INTERP
NOSTOP  HDRYBOT  LIMOP  MINTHICK

The following line is required if  IPESTINT is set to 1.
PESTCTLFILE

All of the following lines are required only if INTERP is set to 1.
NUMPARTYPE  NUMARRAY

Repeat the following line NUMPARTYPE times
PARTYP  INTERPTYP  MININTERPVAL MAXINTERPVAL (LAYER(I), I=1,NLAY)
or
PARTYP  INTERPTYP  MININTERPVAL MAXINTERPVAL (PERIOD(I), I=1,NPER)

Repeat the following two items NUMARRAY times
ARRAYNAME
ARRAY
```

**Figure 18. Specifications of an ASP input file.**

An example of an ASP input file is shown in Figure 19. As is apparent from a comparison with Figure 18, this file is much shorter than it has the potential to be. This is because no variables are supplied in this file governing alterations to MODFLOW-2000 parameter manipulation functionality as INTERP is set to 0. Such alterations are normally required only when spatial parameter definition is undertaken through the use of pilot points; hydraulic property values estimated at these points are spatially interpolated to the remainder of the model domain using kriging. See Doherty (2001) for more details.

```
1    0                        IPESTINT  INTERP
1    0    1   0.00000         NOSTOP    HDRYBOT  LIMOP   MINTHICK
 pestrun.pst                  PESTCTLFILE
```

**Figure 19. Example of an ASP input file.**

The role of each of the variables appearing in Figure 18 is now discussed in turn.

*IPESTINT*

Set this variable to 1 to activate the PEST interface, ie. to direct MF2KASP to write a file containing the derivative of every model-generated observation-equivalent with respect to every adjustable parameter. If IPESTINT is set to 1, then the name of the PEST control file pertaining to the current case must be supplied as the value of the variable PESTCTLFLE. MF2KASP reads this PEST control file to ascertain the name of the file to which it must write its derivatives for PEST to read.

*INTERP*

If INTERP is set to 1, functionality by which calculation of MODFLOW hydraulic property arrays is undertaken on the basis of parameter values is modified in order to allow MODFLOW-2000 to undertake such calculations in a manner that is more appropriate for use with a pilot-points parameterisation scheme. See Section 5.5.2 for further details.

*NOSTOP*

As was explained above, it will be necessary to set the HCLOSE variable governing operation of MODFLOW's solvers to a lower value than you otherwise would if PEST calculates MODFLOW derivatives using finite differences. This is because numerical precision is rapidly lost as two large numbers are subtracted to yield a smaller one as part of the derivatives calculation process; hence those numbers must be calculated as accurately as possible With the tighter convergence setting, it is possible that there will be certain parameter sets which PEST asks MODFLOW to use on particular model runs for which MODFLOW will experience convergence difficulties, particularly if cell drying/re-wetting is operative within the model domain. If solution convergence fails to occur during any time step, MODFLOW ceases execution immediately, failing to proceed to the next time step. When PEST then attempts to read MODFLOW's *observation* process output file (ie. the "*._os*" file) for the model-generated counterparts to field measurements, it will not find it. It will then cease execution with an appropriate error message; the parameter estimation process will then have come to an untimely end.

A similar problem can occur even if derivatives are calculated by MODFLOW-2000 rather than by PEST. MODFLOW-2000 will not write a derivatives file for PEST to read if there is a convergence failure in its iterative derivatives calculation process. PEST execution will then cease when it cannot find the expected file.

If the value of NOSTOP is set to 1, MF2KASP will continue execution even if convergence failure has been experienced during a particular time step, thus preventing occurrence of the problem outlined above. Alternatively, if NOSTOP is set to 0, MODFLOW's behaviour will be unaltered from that of the native MODFLOW-2000. A setting which is neither 0 nor 1 will result in termination of MF2KASP execution with an appropriate error message.

The suggested setting for NOSTOP is 1 when MF2KASP is being used with PEST, whether derivatives are calculated by PEST using finite differences or by MF2KASP using internal MODFLOW-2000 derivatives calculation functionality. However the user is advised to check for MODFLOW mass balance errors, and for other signs of aberrant MODFLOW behaviour, just to be sure that in failing to terminate execution, MF2KASP is not ignoring a significant internal numerical problem.

*MINTHICK*

Normal operation of MODFLOW is such that if the head in a cell falls below the base of that cell, MODFLOW de-activates that cell. However in many instances this may not reflect reality. If the water table falls below the base of the lowest model layer, a limited amount of water may still be able to flow through the rocks underlying the aquifer, a phenomenon that cannot be simulated if all cells above the base of the aquifer are de-activated. Furthermore, many aquifers have an "uneven bottom". Just because the water table falls below the notional bottom of a basal layer (the elevation of the bottom of each cell, which is often poorly known, being assigned to the centre of the cell), this does not mean that groundwater flow is excluded from the entirety of the cell; for example some residual flow may still take place through the lowest parts of the cell.

If the value supplied for MINTHICK in the ASP input file is greater than 0.0, then the modified MODFLOW will not let any bottom-layer cells be de-activated. It will happily allow the MODFLOW-computed head for such cells to fall below the notional bottom of these cells. However when this occurs, intercell conductance will be calculated on the assumption that a saturated thickness equal to MINTHICK remains in the cell. Thus each affected cell remains in contact with its neighbours, and is thus able to conduct water to a limited extent in accordance with the local groundwater gradient, even though the head in that cell is below its base.

Experience has demonstrated that a suitable setting for MINTHICK in many situations is about 1.0m; however this will vary with the aquifer being modelled, and with the length units employed by the model. If MINTHICK is set too small, MODFLOW will experience solution convergence difficulties. If it is set too high, then the bottom layer of the model will effectively become a "confined" layer in the sense that transmissivity no longer varies with water level.

Use of a positive MINTHICK setting is particularly helpful when dealing with large, regional-scale groundwater models. Where such a model is comprised of only a single model layer, a positive value of MINTHICK prevents the occurrence of dry cells altogether. This can have a hugely beneficial effect on the operation of the model for two reasons. Firstly, there will be no numerical errors of the type discussed above in the calculation of finite-difference derivatives incurred through cell de-activation/re-activation. Secondly, MODFLOW boundary conditions and inputs cannot get temporarily or permanently "switched off" in those parts of the model domain where cells would otherwise be de-activated. Nevertheless, caution must always be exercised in setting MINTHICK greater than zero, especially where a substantial amount of pumping takes place from an aquifer. If a pumped bore is situated in a cell where the water table falls below the base of the aquifer (this often being the reason why water levels dropped this far in the first place), the head in that cell can become grossly negative due to the continued action of the pump, and because of the low conductance that exists between that cell and its neighbours.

If a model is calibrated using a positive value of MINTHICK, it is best to retain this positive setting when using the model to make predictions, for then its beneficial effects will be felt under predictive conditions to the same extent as they were felt under calibration conditions. However the user should be aware that problems may be encountered if using the output of such a model with the *advection* process of MODFLOW-2000 or with MT3DMS, for neither of these programs is very tolerant of water levels in an unconfined layer which fall below the base of that layer.

While the use of a positive value of MINTHICK brings with it the advantages already outlined, these benefits do not come without a cost. In the present version of MF2KASP, the calculation of sensitivities has not been corrected for the use of this device. Hence MODFLOW-calculated derivatives cannot be used by PEST where the water level falls below the base of a layer. As a result, PEST cannot use MODFLOW-calculated derivatives if MINTHICK is greater than zero. *Thus IMDERCALC should be set to 0 when using MF2PEST to generate a set of PEST input files if it is intended to provide MF2KASP with a positive MINTHICK setting.*

*HDRYBOT*

Cells can be prevented from going dry through use of the MINTHICK variable only in the lowest layer of the model grid. MF2KASP has no capacity to prevent the occurrence of dry cells in upper layers of the grid.

If an observation bore falls within a cell that goes dry, the *parameter estimation* process of MODFLOW-2000 temporarily removes affected measurements pertaining to that bore from the calibration process. This can have a deleterious effect on the parameter estimation process because it can create a discontinuity in the calculation of the objective function.

When MODFLOW calibration is undertaken using PEST, a different mechanism must be used to handle this situation because, as discussed above, PEST cannot temporarily bring particular measurements in and out of the inversion process. If a bore is situated in a cell that goes dry during at least one model run undertaken during the calibration process, measurements made in that bore must be retained as part of the calibration dataset. However there is a potential for serious problems in retaining such measurements because when a cell is de-activated MODFLOW assigns that cell a dummy head value of HDRY (a user-supplied MODFLOW control variable). In order to make such cells "highly visible", HDRY is often assigned a value of $10^{30}$. Naturally if this head is then assigned to an observation bore and passed to PEST, the objective function will skyrocket; damage to the parameter estimation process will be irreparable.

To prevent this from occurring, MF2KASP provides the option of assigning de-activated cells a head equal to the elevation of the bottom of the cell (instead of HDRY). This will occur if the HDRYBOT variable contained in the ASP input file is set to 1. Heads interpolated to bores in dried out cells will therefore be approximately equal to the elevation of the base of the aquifer at the sites of those bores.

As a further measure, when HDRYBOT is set to 1 the sensitivities of all parameters with respect to affected measurements in the dried out bore are assigned a value of zero. Hence these measurement do not have an effect on the parameter estimation process when the bore is dry. In this respect the handling of dry cells has a similar effect on the parameter estimation process in MF2KASP as it does in the normal MODFLOW-2000.

If HDRYBOT is set to 0, MF2KASP maintains normal MODFLOW functionality. assigning dry cells a value of HDRY.

**HDRYBOT should always be assigned a value of 1 when MODFLOW is used with PEST.** The reason for this is that when HDRYBOT is set to 1 a number of other slight changes are made to normal MODFLOW-2000 operations. None of these affect the manner in which MF2KASP calculates heads, only the way in which observations in dry cells are handled. Normal MODFLOW-2000 operations are such that borehole heads are not recorded in the "._*os*" file (which PEST reads) when the cells containing the bores have dried up; this has disastrous consequences when PEST comes to read this file. However when HDRYBOT is set to 1, heads are recorded for these bores (based on cell heads equal to the base of the layer) in the "._*os*" file. Also, observations pertaining to head-dependent flow boundaries such as drains, rivers and streams, are no longer omitted from the "._*os*" file when respective cells dry out. They are assigned a value of zero and are retained in the "._*os*" file (and they are assigned a sensitivity of zero).

If HDRYBOT is not set to 1 when running MF2KASP with PEST, you may encounter an error message such as the following issued by PEST:-

```
**********************************************************************
Error condition prevents continued PEST execution:-

Unexpected end to model output file MF2KOUT._OS.
Instruction line follows -
"l1 !obore4b!"
**********************************************************************
```

This results from the fact that the "._*os*" file has less observations recorded in it than PEST expects. This will most often be a result of the fact that cells containing observation bores have dried out; also, in some circumstances, it can indicate that the water level has fallen below a group of cells comprising a *drain* observation location. In either case the problem can be rectified if HDRYBOT is set to 1.

On some occasions of MF2KASP usage you may notice a significant difference in the MODFLOW-calculated objective function depending on whether HDRYBOT is set to 0 or 1. This difference is a result of the occurrence of dry observation bores; measurements in such bores are ignored if HDRYBOT is set to 0 (normal MODFLOW behaviour) but used in the computation of the objective function if HDRYBOT is set to 1. It should be noted, however, that in spite of the fact that a HDRYBOT setting of 1 can raise the objective function, it does not have a deleterious effect on the calculation of optimised parameter values.

Depending on the way in which they recognise dry cells, a HDRYBOT setting of 1 may effect the operation of some MODFLOW graphical user interfaces (GUIs) if these GUIs use the expected HDRY value to identify dry cells. However this is not a serious concern because HDRYBOT can be re-set to 0 for normal MODFLOW operation after the parameter estimation process is complete. A model run based on optimised parameter values can then be undertaken before allowing the GUI to read model results.

*LIMOP*

"LIMOP" stands for "Limited Output".

When PEST calculates derivatives of MODFLOW outputs with respect to adjustable parameters using finite parameter differences it needs to run MF2KASP repeatedly (at least once for every adjustable parameter during every optimisation iteration). If, on every occasion that it runs MF2KASP, MF2KASP writes an extensive set of output files, this will result in a lot of unnecessary disk input/output. Where model output files are large (as can happen where many observations are involved in the parameter estimation process), the writing of these files can be a time-consuming procedure.

When LIMOP is set to 1, MF2KASP will refrain from writing the following types of files no matter what input/output settings are provided in the MODFLOW2000 input dataset:-

- all *observation* process output files except for the "*._os*" file (which PEST reads in order to ascertain model-calculated observation equivalents),
- all *sensitivity* process output files,
- all unformatted head, budget and drawdown files.

It is recommended that LIMOP be set to 1 when MF2KASP is run under the control of PEST as part of the model calibration process. Though the benefits of reduced disk I/O are greatest when derivatives are calculated using finite differences (ie. IMDERCALC was set to 0 when running MF2PEST), they can still be substantial when derivatives are calculated by MF2KASP even though the number of MF2KASP runs required to carry out the parameter estimation process is substantially reduced.

However, after the inversion process has reached completion, be sure to set LIMOP back to 0 before running MF2KASP using optimised parameter values and reviewing the calibration results with your GUI. (If you fail to carry out this final model run with a LIMOP setting of 1, you may inadvertently read MODFLOW results from a previous run, undertaken before parameters were optimised.)

*5.5.2 Interpolation Settings*

Figure 20 shows part of an ASP input file in which the INTERP variable is set to 1. A description of the variables required in this file when INTERP is set to 1 are presented herein for the sake of completeness. However for a full discussion of the significance of all of these variables, see Doherty (2001).

```
0    1                          IPESTINT  INTERP
1    0    1   0.00000           NOSTOP    HDRYBOT  LIMOP   MINTHICK
3     2                         NUMPARTYPE NUMARRAY
 HK         1  ARRAY1           1.0e2           1  1  0
 SY         0  0.05             ARRAY2          0  0  1
 RCH        0  1.0E-6           1.0e-3          1  0
 ARRAY1
   1.000000  1.000000  1.000000  1.000000  1.000000
   1.000000  1.000000  1.000000  1.000000  1.000000
   1.000000  1.000000  1.000000  1.000000  1.000000
    .
    .
 ARRAY2
   0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20
   0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20
   0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20  0.20
     .
     .
```

**Figure 20. An ASP input file for which INTERP is set to 1.**

If INTERP is set to 1, then values must be supplied for a number of variables which govern modifications to MODFLOW array-computation on the basis of parameter values. These variables are now described.

*NUMPARTYPE*

NUMPARTYPE is the number of parameter types referenced in the following lines of the file. Each parameter type cited in this file must be suitable for use by the MODFLOW-2000 *LPF* or *RCH* package; possible types are 'HK', 'HANI', 'VKCB', 'VK', 'VANI', 'SS', 'SY' and 'RCH'.

*NUMARRAY*

NUMARRAY specifies how many arrays are listed in the ASP input file. These arrays, if present, specify cell-specific array upper and lower bounds after spatial interpolation has been carried out using kriging factors encapsulated in MODFLOW multiplier arrays.

*PARTYP and INTERPTYP*

NUMPARTYPE lines of data must follow the line in which NUMPARTYPE and NUMARRAY are specified, each containing 4 + *nlay* items or 4 + *nper* items (depending on the parameter type), where *nlay* is the number of layers pertaining to the current model and *nper* is the number of model stress periods. The first entry on each of these lines (PARTYPE) is the name of an LPF parameter type or a recharge parameter type (of which the only type is 'RCH'). The next entry (INTERPTYP), an integer, must be 1 or 0. If it is 0, then internal MODFLOW array construction is undertaken by summing parameter values multiplied by elements of the multiplier arrays in the normal MODFLOW-2000 manner. However if it is supplied as 1, then the elements of property or input arrays are calculated by summing the logs of parameter values, and then raising the summation to the power of 10. Where multiplier arrays are generated by FAC2MF2K on the basis of kriging factors, this is equivalent to kriging on the basis of log parameter values.

*MININTERPVAL and MAXINTERPVAL*

The next two entries on these lines, (viz. MININTERPVAL and MAXINTERPVAL) can be either numbers of names. If the first of these entries is a number, then it represents the minimum value that the pertinent hydraulic property can take over the area of the model domain. However if the entry is not a number then it is assumed to be an array name. In this case minimum hydraulic property values are cell-specific and are read from the named array. Similar considerations apply to MAXINTERPVAL, which provides the maximum allowable interpolated hydraulic property value.

*LAYER(I) or PERIOD(I)*

Each of the final *nlay* entries (if an LPF parameter type) or *nper* entries (if a RCH parameter type) must be either 1 or 0. Each entry pertains to a particular model layer or stress period; note that the counting of layers begins at the surface. If the entry for a particular layer is 0, then data provided previously on this line is inapplicable to that layer; if it is 1, then values supplied for INTERPTYPE, MININTERPVAL and MAXINTERPVAL are all applicable to that layer. The same holds for stress periods in the case of the RCH parameter type.

Note that different lines within the first section of the file can cite the same property type (ie. PARTYP). This may occur if the interpolation specifications are different for different layers (LPF parameters) or different stress periods (the RCH parameter).

*ARRAYS*

Then follow NUMARRAY arrays. The dataset provided for each array is comprised of the array name, followed by the array itself. Note that the normal MODFLOW protocol for array input is not followed, in that each array is not preceded by a header line in

which formatting specifications are provided. Thus the array cannot be read using the MODLFOW array reading utility subroutine U2DREL.

## 5.6 Running MF2KASP in Parameter Estimation Mode

It is best to disable all ASP functionality when running MF2KASP in parameter estimation mode; thus it runs as the normal version of MODFLOW-2000. Hence MINTHICK and INTERP will be set to 0 (sensitivities would be in error if these were not thus set), and HDRYBOT will be set to 0 (alterations made to the calculation of the objective function with HDRYBOT set to 1 are not compatible with MODFLOW-2000 parameter estimation functionality). Also, because LIMOP is set to 0, no MODFLOW-2000 output will be disabled.

# 6. References

Doherty, J., 1994. PEST: Model-Independent Parameter Estimation. Watermark Computing.

Doherty, J., 1995. PEST Utilities for MODFLOW and MT3D Parameter Estimation. Watermark Computing.

Doherty, J., 2000. PEST-ASP. Model-Independent Parameter Estimation. Watermark Numerical Computing.

Doherty, J. 2001. Manual for Groundwater Data Utilities. Watermark Numerical Computing.

Harbaugh, A.W., Banta, E.R, Hill, M.C. and McDonald, M.G., 2000. MODFLOW-2000, The U.S. Geological Survey Modular Ground-Water Model - User Guide to Modularization Concepts and the Ground-Water Flow Process. U.S. Geological Survey Open-File Report 00-92. Reston, VA.

Hill, M.C., Banta, E.R., Harbaugh, A.W. and Anderman, E.R., 2000. MODFLOW-2000, The U.S. Geological Survey Modular Ground-Water Model - User Guide to the Observation, Sensitivity and Parameter-Estimation Processes and Three Post-Processing Programs. U.S. Geological Survey Open File Report 00-184.

Zheng, C. and Wang, P. P., 1998. MT3DMS Documentation and User's Guide. Departments of Geology and Mathematics. University of Alabama.